RC 6341 (#26178) 6/15/76 (Rec'd 12/30/76)
Computer Science 12 pages

Hybrid Analysis/Simulation: Distributed Networks

C.H. Sauer, L.S. Woo

IBM Thomas J. Watson Research Center
Yorktown Heights, New York 10598

W.Chang

IBM Data Processing Division
White Plains, New York 10604

(4D) 12p.

Abstract: Hybrid solution techniques combining analysis and simulation can be used for complex models when analytic solutions are intractable and simulation solutions are unwieldy. Models of distributed teleprocessing networks are often complex since processing capability is distributed at several nodes instead of having all processing done at a host computer. A model has been developed to study some of the characteristics of the Systems Network Architecture (SNA), Level 2, and related software. This model is solved using hybrid techniques.

LIMITED DISTRIBUTION NOTICE

This report has been submitted for publication elsewhere and has been issued as a Research Report for early dissemination of its contents. As a courtesy to the intended publisher, it should not be widely distributed until after the date of outside publication.

Copies may be requested from:
IBM Thomas J. Watson Research Center
Post Office Box 218
Yorktown Heights, New York 10598

1. INTRODUCTION

The primary purpose of this paper is to report on preliminary efforts at combining analytic and simulative methods to develop performance evaluation approaches which will be appropriate to configuration of current and future distributed networks. We refer to this approach as a hybrid of analyis and simulation, or briefly, as hybrid simulation. In the remainder of the paper, we will mainly discuss those aspects of distributed networks which we believe cannot be dealt with analytically. We will characterize these aspects in more detail. Then we will discuss how they may be dealt with by hybrid simulation. Finally, we discuss results from a sample system model which we have constructed.

Teleprocessing networks have allowed many computer users to simultaneously have individual access to computers. One approach to teleprocessing is the *centralized* network, where a central computer system is directly connected to terminals by communication links. The central system is responsible for all of the user computation and is also responsible for handling the communication links and terminals. The centralized approach has several problems:

1. The system may spend a considerable portion of its resources on communicating messages associated with minor user activities.
2. The major portion of the response time seen by minor user activities will be the time messages spend in the communication links.
3. If the central computer fails, then all of the terminals fail.
4. The communication links may be dramatically under-utilized. The communication links may be rather expensive.

An alternative approach is the *distributed* network. In a distributed network, processing capability, either for communication or user purposes, is available at more than one location. For the purposes of this paper, we will focus on hierarchical distributed networks, where a central processing facility is augmented by processors at other locations.

The communication processing in a distributed network will be handled by processors distinct from the central system. The communications processors may be specialized hardware or small general purpose machines. A communications processor will have a relatively high capacity path to the central system or another processor higher in the hierarchy, and several lower capacity paths to processors or terminals lower in the hierarchy. Some user activities may also be handled in processors distinct from the central site. The processors handling such activities may or may not be distinct from the communications processors.

For example, a chain of retail stores may have sales terminals which perform the functions of cash registers. These sales terminals may be connected to computers to maintain inventory records. A centralized network would be inappropriate for this application because of the expense of communicating each sale to the central system, because of the time the salesperson and customer might have to wait, and because of the inventory system failing when the central system failed. A distributed network would likely have small processors in each store. The sales terminals would normally only deal with the store processor, and the store processor would periodically update the inventory information at the central system. Depending on the number of stores and other processing the stores might request, there would be one or more additional processors which would interface the central system and the communication links. See figure 1.

Thus distributed networks have advantages with respect to efficiency, with respect to availability, and with respect to function in that they make available applications which would be infeasible with a centralized network. However, distributed networks are more difficult to design and configure than centralized networks. One must not only provide adequate resources, one must distribute the resources appropriately. Since communication resources are being shared, consideration must be given to how the sharing is to be controlled. Disciplines used by low capacity communication links may be inappropriate for high capacity links.

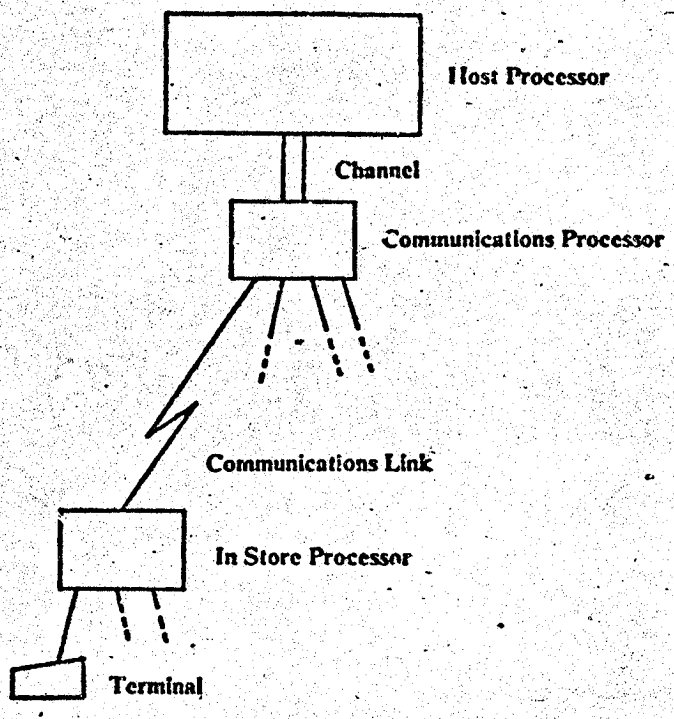

Figure 1 - Distributed Teleprocessing Network

The above design considerations have lead to segmentation of messages so that each segment on a communications link will have a maximum length, to memory dedicated for use as buffers, and to protocols to control traffic flow in the network. (The term segment may have system specific connotations which we do not desire. We use segment in a generic sense, with approximately the same meaning as "block" or "packet.")

The distribution of capability plus the added design characteristics have made it difficult to apply analytic techniques for the configuration of centralized systems to the configuration of distributed systems. Though these techniques may be appropriate to each individual processor in a distributed system, they do not capture the interaction among components. This is especially true when buffer capacity is constrained and protocols are used to prevent buffer overflow.

On the other hand, a strictly simulative approach, even a fairly gross level approach, is likely to become unwieldy for moderate to large networks. If an unwieldy simulator is used, then it will probably be too expensive to hope for adequate statistical analysis of simulation results.

2. BUFFER CONTENTION AND RELATED CONTROL MECHANISMS

When a message is sent from one node of a network to another, the receiving node must have a buffer area to store the incoming message. The transmitting node also has a buffer holding the message, and it holds the message at least until the receiving node acknowledges receipt. If no such acknowledgement is made within a timeout period then the transmitting node will retransmit the message. The lack of acknowledgement may either occur because of a transmission error or because the receiving node ran out of buffers.

The above assumes that the buffers are always in primary storage (communications processors frequently do not have any secondary storage). Assuming that secondary storage is not available for buffering, it is clear that contention for buffers has much potential for impacting network performance. So it is important to include mechanisms in a network to try to ease buffer contention in general and to prevent nodes from running out of buffers. Even networks that have sufficient buffers for normal operation may have buffer shortages when the network or a subnetwork experiences an unusual surge of traffic.

One control mechanism already mentioned is to segment messages so that there is a maximum, relatively small size for a segment. The immediate gain from this is that the maximum buffer size required is known.

A second control mechanism is to limit the flow from a node at one end of a path to a node at the other end. Such a control mechanism must be coordinated with all the nodes on the route and may regulate the flow out of such nodes. The IBM Systems Network Architecture (SNA) uses "pacing" algorithms to effect such control [1]. Pacing limits the number of messages which can be sent to a node before that node requests that more messages be sent.

A third control mechanism is for a node to notify other nodes that is is running out of buffers and does not wish to receive further segments until it frees some of its occupied buffers. There are two potential problems with this mechanism: the node may need to receive control messages, such as pacing or acknowledgement responses, in order to free buffers, and the mechanism may cause other nodes to run out of buffers and/or cause deadlocks. The Network Control Program (NCP) for the IBM 3704/3705 communications processors has such a mechanism, called "slowdown" [2].

These control mechanisms clearly can affect each other. The difficulty of dealing analytically with buffer contention and control mechanisms forces one to consider a simulation approach. Other control mechanisms may also be used with distributed networks. Though these mechanisms are not considered here, the techniques we use can be applied to other control mechanisms as well.

3. HIERARCHICAL/HYBRID SIMULATION

3.1 Hierarchical Models

The use of hierarchical models, i.e. submodels included in a hierarchy to result in a total system model, is fairly recent. Hierarchical modeling can be done exactly for certain queueing networks [3] and is the basis for many analytic approximations for queueing networks [4,5]. Simulative submodels have been also used in an approximate solution of a model of a large data base system [6]. Though the potential advantages of hierarchical models are quite attractive, it should be kept in mind that hierarchical techniques are not well understood. Further, there is little hope for error bounds or other theoretical results since one is applying the techniques because analytic or numerical solutions are not obtainable. The main hope for understanding is in empirical efforts such as those already referenced and those described here.

Besides accuracy, which must remain a consideration throughout, the main questions in hierarchical modeling are

1. How does one define the submodels?
2. How does one interface the submodels to construct the hierarchical model?
3. How does one solve the submodels and the overall model?
4. Is the hierarchical approach appropriate? (Is there any advantage over a strictly simulative approach?)

These questions may not be fully answerable, but possibilities can be proposed.

In a distributed network, one can try to define submodels which are more or less independent of each other except for input and output traffic. For each processor there may be a submodel,

with submodel complexity corresponding to processor complexity. There may also be submodels for communication links, for buffer contention and for control mechanisms.

The interface between models must both be detailed enough to capture dynamic behavior and yet simple enough to preserve the advantages of hierarchical modeling. The referenced previous hierarchical modeling was based on queueing network models and used the results of [3] in an approximate fashion. Our efforts are also based on generalized queueing networks [7,8,9]. In the overall model we represent a submodel by a queue with queue length dependent service rate. The throughput through the submodel, for a given population, is used as the service rate for queue length equal to that population. This may be refined to consider distinctions between queue lengths and throughputs for different kinds of messages and/or segments.

Usually one will solve the submodels by the least expensive technique which can be applied. This results in most of the submodels being evaluated by some analytic or approximate method. The least tractable ones will be dealt with by simulation. In the case of distributed networks, the characteristics described in section 2 are the likely candidates for simulation.

The tradeoffs in hierarchical/hybrid simulation are not all clear. If one can replace a complex subsystem simulation by a state dependent queue, then it seems one should do so. But simulation of a state dependent queue is non-trivial and may have cost equal to or greater than the cost of simulating a simple subsystem. We are left with the question of how to define "simple" and "complex."

3.2 Submodel Characterization and Solution

In this subsection we describe the submodels we have studied and the solution techniques employed. In section 4 we describe a sample system study using these submodels.

3.2.1 Host Subsystem. When a DB/DC (data base/data communication) application operates, the CPU serves many partitions or regions with a non-preemptive priority scheduling discipline. The partitions are for the application programs and the system software resources. The system software has higher priority. In general there are several types of transactions in the application programs which may be ordered by priority. The number of active transactions of each type is usually held below some maximum value. The CPU service is only interrupted by I/O requests for an access to a file or for paging purposes.

To simplify our analysis of the host subsystem, the I/O holding time, i.e. the sum of the waiting and service times, is computed independently. The I/O submodel takes account of the contention of channels between devices. The CPU is treated as a single server which serves customers of several classes of finite population. The customers are the different transactions alternating between application and system processing. See figure 2. Each customer gets its holding time before rejoining the CPU queue. Each transaction completes a finite chain of alternate CPU and I/O services and leaves the host subsystem. The transaction processing time is defined as the time for completing the finite chain. The throughput for a given population of transactions is that population divided by the transaction processing time. Our problem is to define an analytical model for evaluating the transaction processing time.

The above model may be solved by one of the following analytic techniques:

1. Local balance solution [10] in which all the priorities are ignored but the chain structure is preserved.
2. A semi-Markov solution [11] where the priorities are observed but the CPU service chain is lumped together into a weighted average with hyper-exponential distribution. This method at present is feasible for two classes of transactions.
3. Iterative queueing network analysis [12] where the solution for many classes of transactions can be approximated. However, the chain structure is represented by the lumped average of the CPU service times and the lumped service time is assumed to have exponential distribution.

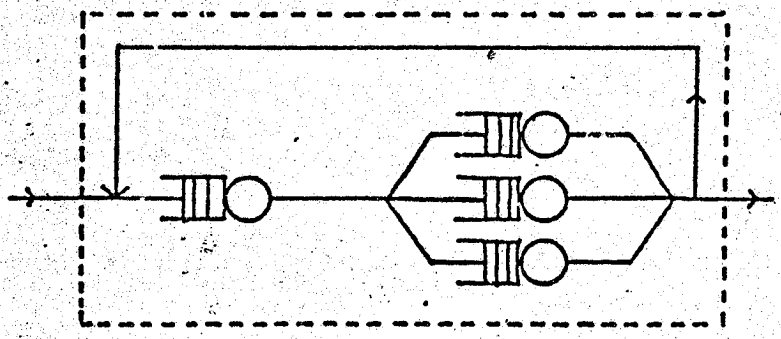

Figure 2 - CPU-I/O Subsystem
(Limited Populations of Each Type of Transaction)

In order to establish the accuracy of the above solutions, we compared the results with simulation results from APLOMB [7,8]. (APLOMB is a simulation package for a generalized class of queueing networks; it provides facilities for determining confidence intervals for simulation results and sequential sampling techniques for determining appropriate simulation run lengths.) The host subsystem of DB/DC is simulated with random arrivals of transactions. The simulation takes into account the priorities of the different partitions. The average transaction processing time and the transaction queue distribution are obtained.

For the analytic solutions, we compute the transaction processing time for every possible combination of transaction types up to the maximum allowable values. The average processing time is evaluated as a weighted sum with the weights taken from the simulation transaction queue length distributions. The results of a sample problem with two transaction types is shown in figure 3.

CPU Utilization	Priority	Simulation	Method 1	Method 2	Method 3
.282	None	.254	.245	.242	.240
.563	None	.344	.344	.334	.330
.776	None	.693	.706	.687	.683
.282	High	.244	.243	.238	.235
.563	High	.286	.318	.295	.289
.776	High	.364	.454	.381	.392

Figure 3a - Short Transactions
Host Response Times in Seconds

CPU Utilization	Priority	Simulation	Method 1	Method 2	Method 3
.282	None	1.10	1.08	1.15	1.14
.563	None	1.35	1.29	1.47	1.44
.776	None	1.67	1.55	1.81	1.77
.282	Low	1.18	1.07	1.17	1.16
.563	Low	1.69	1.28	1.59	1.57
.776	Low	2.98	2.03	2.66	2.60

Figure 3b - Long Transactions
Host Response Times in Seconds

Comparison of the results shows that when the transactions are of the same priority, all three solution techniques compare well with the simulation. However, when the two transaction types have different priorities, the results from solution 1 (the local balance solution) differ significantly from the simulation at high CPU utilizations. The other solutions give good results for this case.

3.2.2 Communication Links We assume that the communication link is full duplex SDLC (Synchronous Data Link Control) [13]. The approach can also be applied to other types of links. Multi-point cluster control units on each link are polled cyclically. (Cluster control units in SNA are remote processors which perform communications functions for a cluster of terminals or other devices. Cluster controllers are also used for application processing [14].) At each poll inbound messages may be transmitted to a communications controller. In SDLC there is an upper bound on the number of segments that may be transmitted at a single poll. Similarly, the number of outbound segments per transmission to a cluster control unit is bounded. Also, the outbound message shares the output line with the polling messages and has lower priority.

Communication link analysis under simplifying assumptions has been investigated by many authors [15,16,17]. These assumptions are that traffic arrives at the queue randomly and that all the messages in a given queue will be transmitted after a single poll. The multi-point polling problem can be analyzed for a cyclic order service with random switching times between queues. The switching time represents the polling message transmission time. Rigorous methods take into consideration the interdependence of arrivals and services between queues and result in complex Laplace transforms. It is difficult to evaluate the transform for the mean waiting time when the number of queues is greater than two.

Approximate methods based on further simplifying assumptions also exist [18,19]. The analysis takes a much simpler form. The results from this approach compare well with rigorous analysis for two queue systems, provide the traffic among queues is approximately equal. The nature of the independence assumption is such that the *largest* error occurs in a two queue system. The error decreases with the number of queues.

For the representation in hybrid simulation, we adopted the approximate method. However, the method is modified to include the effect of polling wait at the output line and the effect of the upper bound of the number of segments per transmission. This modified method was developed by Y.B. Chen of the IBM DPD [20].

3.2.3 Segmentation of messages

The simulation package APLOMB provides constructs for one customer in a queueing network to become several related customers and for several such related customers to become one customer again [7,8,9]. One of the intentions of these constructs was representation of segmentation of messages, so this mechanism fits easily into the simulation.

3.2.4 Pacing

Pacing as used in SNA has two parameters: N, the number of messages which can be sent without a response, and M, the number of the one of the N messages which is to be marked for a pacing response [2]. One can think of the receiving node as having 2 sets of N buffers. (Actually the receiving node need have only 2N−M buffers.) The purpose of the second parameter is to anticipate when the buffers will be available. Since it will take some time for the pacing response to reach the sending node, having $M < N$ allows potential overlap of the freeing of the receiving buffers and the transmission of the pacing response.

The "passive" server construct of APLOMB is well suited to representing pacing [7,8,9]. Passive servers are distinguished from "active" servers (servers in the traditional sense) in that they are acquired as a prerequisite to acquisition of other servers (active or passive). They function to block passage of a customer through the queueing network until a passive server is available. Acquisition is instantaneous, and the passive server is held until explicitly released. It is also possible for customers to destroy and create passive servers.

In representing pacing, we use a pool of N units of a passive server. Before a message can leave a transmitting node, it must acquire a unit of the passive server corresponding to the receiving node. The Mth of N acquisitions is marked for a pacing response. When a message arrives at the receiving node, it destroys its unit of passive server. (If it were to release its unit,

then that unit would immediately be available to a message at the receiving node.) The message which has been marked for a pacing response creates another customer in the queueing network to represent the pacing response and is sent to the transmitting node. When this pacing response customer arrives at the transmitting node, it creates N units of the passive server (the origingal N units have been or will be destroyed) and N additional messages may now proceed. See figure 4.

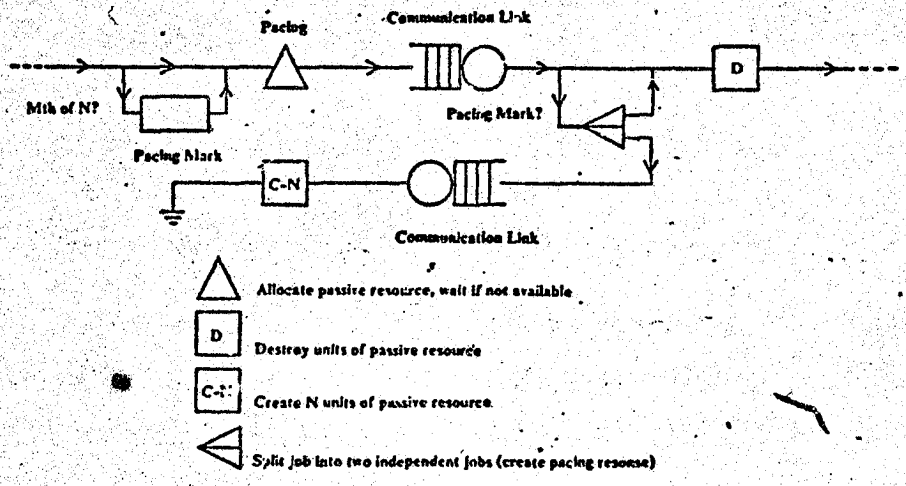

Figure 4 - APLOMB Model of Pacing

3.2.5 Slowdown

Representation of slowdown also makes use of passive servers (it is possible for a customer to simultaneously hold units of several passive servers). Passive servers are used to represent the buffers. When the number of units of the buffer passive server falls below a threshold, then slowdown is entered. There are also passive servers used as controls to indicate whether the node with slowdown control is in slowdown or not. There is such a passive server for each node which may send to the node with slowdown control. When a node is to send a segment to a node with slowdown, it must first first acquire the control passive server. If the control passive server is not available, then the controlled node is in slowdown and the segment must wait for normal operation to resume. Once a segment acquires the control passive server, it immediately releases that passive server and proceeds with transmission to the control node. (Releasing the passive server makes it available for the next segment to test.) When a segment arrives at the receiving node and causes the slowdown threshold to be passed, it generates slowdown messages which are sent to all of the nodes with control passive servers. The slowdown message acquires the control passive server and destroys it. There is a second threshold which determines when the node leaves the slowdown condition. When a segment releases a buffer during slowdown and causes this second threshold to be passed, it generates messages to be sent to each of the nodes with control passive servers. These messages indicate that the nodes may resume transmission. The messages do so by creating a unit of the control passive server. See figure 5. The control passive servers are labelled "SC." There is also a flag passive server used to indicate whether or not the node is in slowdown.

4. SAMPLE SYSTEM STUDY - THE ASA MODEL

A hybrid simulation based upon the above submodels was constructed using APLOMB [7,8] to study the feasibility of the approach and the interactions of the control mechanisms. The hybrid simulation model was named ASA for "Analytic/Simulation Approach." Though the model constructed is based on a class of hypothetical SNA systems, the authors' imperfect knowledge of SNA and the above stated goals led us to take liberties with respect to system details which seem irrelevant to the current work. Before the current model can be useful for performance evaluation of actual systems it will be neccessary to reexamine model assumptions and validate the model with

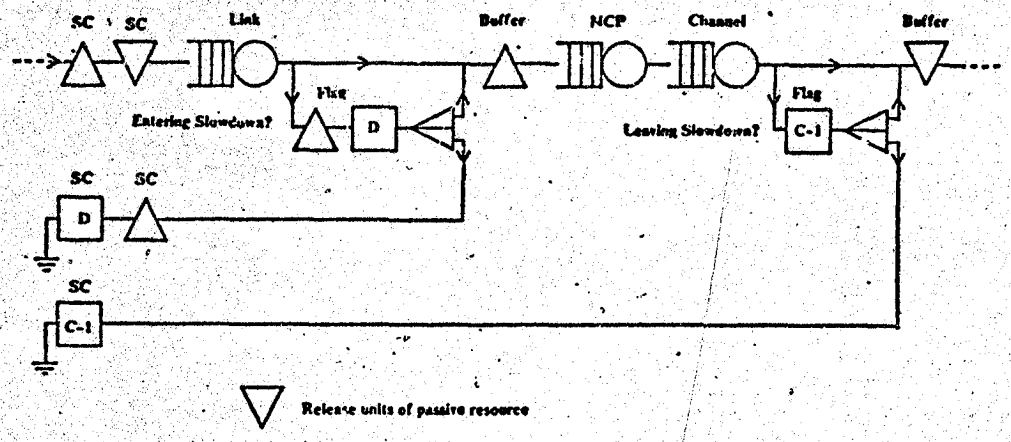

Figure 5 - APLOMB Model of Slowdown

measurements from appropriate systems. It should be emphasized that values chosen for the parameters are not neccessarily those that would be used in an installed system. Some of the parameters used are based on the authors' guesses. Others are intentionally allowed to have atypical values to emphasize the effects we are studying.

An important consideration in any simulation is statistical analysis of the output. APLOMB normally computes confidence intervals using regenerative methods [7,8,21,22]. However, we were unable to apply the regenerative techniques for the large models we simulated. A heuristic technique was used to obtain confidence intervals; a future paper will discuss this technique.

We now present results from parametric studies of a sample system. The system under study consists of four communication links and eleven cluster control units. See figure 6. There are two transaction types, inquiry transactions with small communication and processing demands and inventory transactions with larger demands. Pacing is used only for messages of this second transaction type. The application program at the host is the IBM Customer Information Control System (CICS) [23]. For simplicity, we assume a single segment size throughout the system and that all message lengths are an integral multiple of this segment size. The traffic data and message lengths are shown in figure 7.

The parameters varied here are the number of buffers at the communications controller and the pacing parameters from the host to the communications controller. Pacing parameters from the communications controller to the cluster control units remain fixed. Note that we treat this second pacing as being to a single logical unit within the cluster controller [14].

The performance metrics we examine are the number of communications controller slowdowns in a given time, the number of segments which must be retransmitted because of communications controller buffer depletion and the transaction response times. The transaction response time is defined as the time interval between the arrival at the cluster controller and the return to the cluster controller.

The simulated time for each set of parameters is approximately 83 minutes. The running time for each set of parameters was approximately 2 minutes on an IBM 370/168, so the ratio of running time to simulated time is very small in comparison with conventional simulation approaches. Tabulation of the performance metrics is given in figures 8, 9 and 10. These results indicate the performance improves appreciably when the number of buffers is increased from 32 to

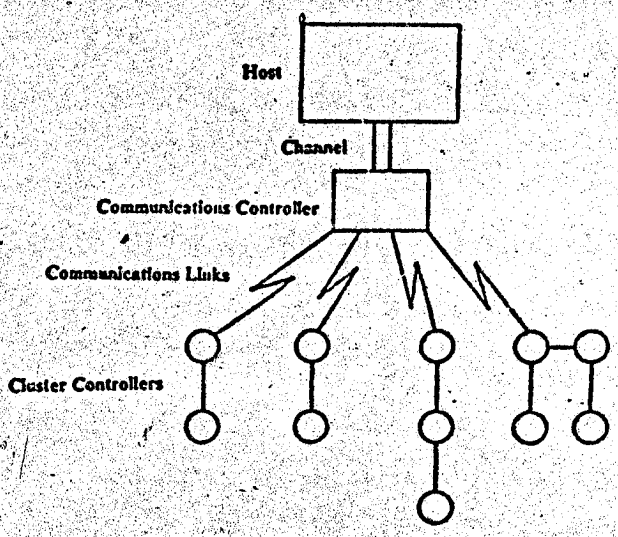

Figure 6 - Configuration of Sample System

Communication Link	1	2	3	4
Link Capacity (baud)	2400	2400	4800	4800
Number of Clusters	2	2	3	4
Messages per hour per cluster				
Inquiry	400	400	720	800
Inventory	50	50	90	100

Message Sizes:
Inquiry -- 1 segment incoming, 1 segment outgoing
Inventory -- 4 segments incoming, 8 segments outgoing

Pacing for inventory messages only

Figure 7 - Sample System Description

48. Note that 32 is an unreasonably small number of buffers and thus the model results should not be taken as indicative of installed system results. When the number is increased again to 64, only a slight improvement is noticed, implying that 48 buffers are adequate. Without pacing from the host, the messages tend to clog the buffer pool and result in more frequent slowdowns. As the pacing is included the performance improves and when the parameters are adjusted the performance improves again. The response times shown are for the first of the four links. Other links showed similar relative behavior.

Number of Buffers	32	32	32
Slowdown Threshold (buffers available)	8	8	8
Resume Threshold (buffers available)	12	12	12
Controller to Cluster Pacing	1,1	1,1	1,1
Host to Controller Pacing	None	1,1	2,1
Number of Slowdowns	466	430	466
Retransmitted Inbound Segments	407	361	390
Retransmitted Outbound Segments	942	835	994
Mean Inquiry Response Time (first link)	3.05	2.85	3.03
Mean Inventory Response Time (first link)	8.40	8.49	8.56

Figure 8

Number of Buffers	48	48	48
Slowdown Threshold (buffers available)	12	12	12
Resume Threshold (buffers available)	18	18	18
Controller to Cluster Pacing	1,1	1,1	1,1
Host to Controller Pacing	None	1,1	2,1
Number of Slowdowns	112	104	112
Retransmitted Inbound Segments	22	13	20
Retransmitted Outbound Segments	81	35	77
Mean Inquiry Response Time (first link)	2.39	2.30	2.35
Mean Inventory Response Time (first link)	7.85	7.73	7.89

Figure 9

Number of Buffers	64	64	64
Slowdown Threshold (buffers available)	16	16	16
Resume Threshold (buffers available)	24	24	24
Controller to Cluster Pacing	1,1	1,1	1,1
Host to Controller Pacing	None	1,1	2,1
Number of Slowdowns	16	15	7
Retransmitted Inbound Segments	2	2	1
Retransmitted Outbound Segments	3	1	2
Mean Inquiry Response Time (first link)	2.25	2.25	2.22
Mean Inventory Response Time (first link)	7.73	7.73	7.73

Figure 10

Since there is no pacing control for inbound messages, we investigated the distribution of message queues at the host. Simulation runs were made with different host processing rates. Figure 11 compares these runs. Again we emphasize that the results for the slower CPU should not be taken as indicative of installed systems.

5. VALIDATION AND OTHER FUTURE EFFORTS

The major lack in our work has been validation. There are two complementary validations required: that the hybrid solution technique gives an accurate solution of our conceptual model and that our conceptual model captures the performance of actual systems.

Some progress has been made on the first aspect of validation. Where analytic submodels are used the analytic results have been compared with more detailed simulation results, doing this with the submodel isolated from the overall model. Some of these comparisons are described in section 3. Of course these comparisons say nothing about the accuracy of the interfaces between submodels nor about the advantages of the hybrid technique relative to a strictly simulative approach.

The current hybrid model includes the capability of using either an analytic solution for the host processing or the simulation host model described in section 3. Comparison runs between these capabilities have shown good agreement between the two. The hybrid version has a modest advantage in processing time. That the gain is less than about 10% is not surprising to us since the simulation of the state dependent server approaches the complexity of our gross level host simulation.

When we constructed the global model we decided not to attempt a similar capability for hybrid/simulation comparisons with the polling submodel. This decision was based on the increase in programming complexity and the assumption that our primary validation would be with measurements from an actual system. So far we have not been able to obtain such measurements and it seems appropriate that we now include the comparison capability for the polling submodel. It is expected that such comparisons would show a much more dramatic processing advantage for the hybrid model.

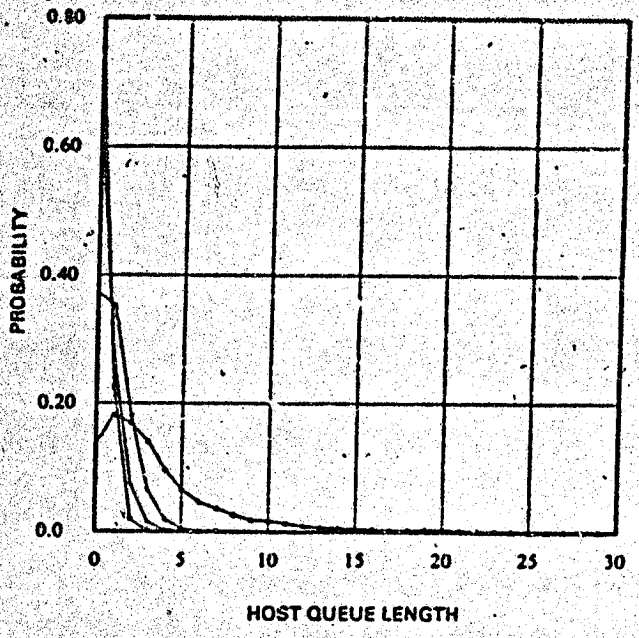

Figure 11
Dot is for inquiry messages and faster CPU.
Triangle is for inquiry messages and slower CPU.
Square is for inventory messages and faster CPU.
Diamond is for inventory messages and slower CPU.

As just stated, we have been disappointed by our inability to obtain measurements for validation purposes. We have not given up on obtaining such measurements and doing this validation. We are also exploring the possiblity of comparing our model with a much more detailed simulation model. In either case, we also need to obtain more information about typical parameter values in installed systems.

Once our current efforts have been properly validated, including the inevitable modifications to our assumptions and sample model, there are several possible extensions. From the beginning we have intended to include an analytic submodel of the cluster controller interaction with the terminals and this should still be done. We would also like to include aggregation of communication links and subnetworks so that it would be feasible to handle much larger networks. Consideration of time dependent arrival patterns may also be appropriate.

In addition to these teleprocessing network considerations, there are also the unanswered questions posed in section 2 as to how hierarchical solutions should be constructed and under which circumstances hierarchical and hybrid techniques are appropriate.

REFERENCES

1. H.R. Albrecht and K.D. Ryder, "The Virtual Telecommunications Access Method: A Systems Network Architecture Perspective," *IBM Systems Journal* 15, 1 (1976).
2. *IBM 3704 and 3705 Communications Controllers Network Control Program Storage and Performance Estimates*, No. GC30-3006-2, IBM Corporation, Data Processing Division, White Plains, New York.
3. K.M. Chandy, U. Herzog, and L.S. Woo, "Parametric Analysis of Queueing Network Models," *IBM Journal of Research and Development* 19, 36 (1975).

4. K.M. Chandy, U. Herzog, and L.S. Woo, "Approximate Analysis of General Queueing Networks, *IBM Journal of Research and Development* **19**, 1 (1975).
5. C.H. Sauer, and K.M. Chandy, "Approximate Analysis of Central Server Models," *IBM Journal of Research and Development* **19**, 3 (1975).
6. J.C. Browne, K.M. Chandy, R.M. Brown, T.W. Keller, D. Towsley and C.W. Dissley, "Hierarchical Techniques for Development of Realistic Models of Complex Computer Systems," *IEEE Proceedings* **63**, 6 (1975).
7. C.H. Sauer, "Simulation of Generalized Queueing Networks," *Proceedings 1975 Summer Computer Simulation Conference.*
8. C.H. Sauer, "Characterization and Simulation of Generalized Queueing Networks," IBM Research Report RC 6057, IBM T.J. Watson Research Center, Yorktown Heights, New York.
9. M. Reiser and C.H. Sauer, "Queueing Network Models: Methods of Solution and their Program Implementation," to appear in K.M. Chandy and R.T. Yeh, editors, *Current Trends in Programming Methodology, Volume III: Software Modeling and Its Impact on Performance*. Prentice Hall (1977).
10. M. Reiser, "QNET4 User's Guide," IBM Research Report RA-71, Yorktown Heights, New York (1975).
11. W.M. Chow, "Central Server Model for Multiprogrammed Computer Systems with Different Classes of Jobs," *IBM Journal of Research and Development* **19**, 3 (1975).
12. E. MacNair and L. Woo, "Iterative Queueing Network Analysis (IQNA) of Closed Queueing Networks and Its Implementation," to appear as an IBM Research Report, Yorktown Heights, New York.
13. R.A. Donnan and J.R. Kersey, "Synchronous Data Link Control: A Perspective," *IBM Systems Journal* **13**, 2 (1974).
14. J.H. McFadyen, "Systems Network Architecture: An Overview," *IBM Systems Journal* **15**, 1 (1974).
15. J.S. Sykes, "Simplified Analysis of an Alternating Priority Queueing Model with Setup Time," *Operations Research* **18** (1970).
16. M. Eisenberg, "Queues with Periodic Service and Changeover Time," *Operations Research* **20** (1972).
17. A. Konheim, "Waiting Lines and Times in a System with Polling," *Journal of the ACM* **21**, 3 (1974).
18. M.A. Leibowitz, "An Approximate Method for Treating a Class of Multiqueue Problems," *IBM Journal of Research and Development* **5**, 4 (1961).
19. J.F. Hayes and D.N. Sherman, "A Study of Data Multiplexing Techniques and Delay Performance," *The Bell System Journal* **51**, 9 (1972).
20. Y.B. Chen, private communication.
21. S.S. Lavenberg and D.R. Slutz, "Introduction to Regenerative Simulation," *IBM Journal of Research and Development* **19**, 5 (1975).
22. D.L. Iglehart, "The Regenerative Method for Simulation Analysis," to appear in K.M. Chandy and R.T. Yeh, editors, *Current Trends in Programming Methodology, Volume III: Software Modeling and Its Impact on Performance*. Prentice Hall (1977).
23. *Customer Information Control System (CICS/VS) General Information Manual*, No. GH20-1280-0, IBM Corporation, Data Processing Division, White Plains, New York.