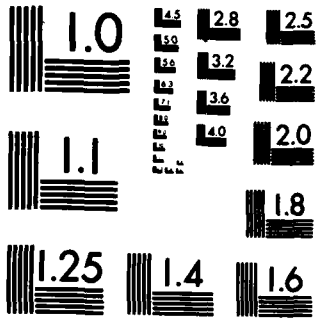
AD-A147 446 DECENTRALIZED CONTROL OF SCHEDULING IN DISTRIBUTED SYSTEMS(U) MASSACHUSETTS UNIV AMHERST DEPT OF ELECTRICAL AND COMPUTER ENGINEERING J A STANKOVIC 1/1

UNCLASSIFIED 14 SEP 84 F/G 9/2 NL

END


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6

RESEARCH AND DEVELOPMENT TECHNICAL REPORT
CECOM

DECENTRALIZED CONTROL OF SCHEDULING IN DISTRIBUTED SYSTEMS

John A. Stankovic
Dept. of Electrical & Computer Engineering
University of Massachusetts
Amherst, MA 01003

Quarterly Report for the Period 15 June 84 to 14 Sept 84

Distribution Statement

Approved for public release;
distribution unlimited

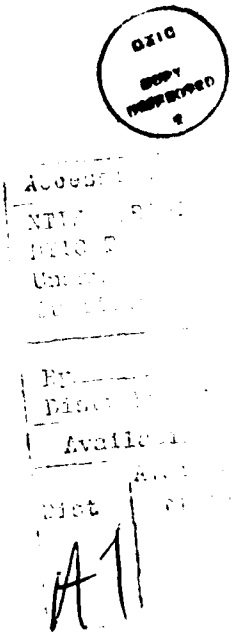
Prepared for
Center for Communications Systems (C. Graff)

CECOM
U S ARMY COMMUNICATIONS-ELECTRONICS COMMAND
FORT MONMOUTH, NEW JERSEY 07703

Table of Contents

1.0 Introduction

This report summarizes the work performed during the period for June 15, 1984 to September 14, 1984. The work concentrated on the stability question of decentralized algorithms, further development of our test-bed simulation program and completion of a Master's thesis. Most new results obtained in this quarter are still preliminary and we expect more complete conclusions by the end of the next quarter.



2.0 Main Text

2.1 Stability

Many distributed scheduling algorithms have been developed to improve the performance and reliability of distributed computer systems. Most of these algorithms have been evaluated analytically or by simulation studies. All of the algorithms make many assumptions and the resulting performance always seems to be "good", subject to their assumptions.

The analytical models assume known averages and distributions for many parameters, and that the system is in equilibrium. Because of this, stability concerns of these algorithms are not addressed.

Algorithms tested by simulation also make many assumptions about averages and distributions, and even about the system being in equilibrium. However, in the simulation study approach, the algorithms are generally more adaptive to changing network conditions and a wider range of conditions is normally tested. Usually, there is an attempt to determine the functional dependence of various parameters on each other. This is sometimes referred to as tuning the parameters of the algorithm. In actuality, the tuning process "implicitly" addresses the stability questions, subject to the assumptions made in the simulation model. For example, by making many runs to choose the interval (rate) at which the scheduling algorithm executes, or to choose the frequency or amount of state information to transmit, a given distributed scheduling algorithm may then perform "well". Performing well is difficult to define but would imply giving good average response time or throughput, and being stable, i.e., doesn't perform too poorly under any reasonable input. The reason that the algorithm performs well under the

conditions (including inputs) tested is that the functional dependence of parameters such as the rate of task arrivals, the characteristics of tasks, the scheduling interval, the accuracy of state information, etc. is "somehow" taken into account., These functional dependences are not constant, so distributed scheduling algorithms must attempt to adapt to changing network conditions. However, it is very difficult to define what is proper under all conditions, to insure that a distributed algorithm always achieves that objective, and to evaluate (or even identify) all possible functional dependencies that exist.

During this quarter we studied two distributed scheduling algorithms one based on stochastic learning automata and the other based on bidding, both of which contain "explicit" mechanisms for dealing with some aspects of stability. For the stochastic learning automata algorithm we showed that it can be made to work well. For the bidding algorithm we showed that typical prediction technique (moving average forecast) does not work, and suggest a possible solution. The solution is that we must take into account the details of the real-time environment. For example, in predicting the future load at a host we need to consider factors relating to local tasks as well as factors relating to the effect of system-wide load balancing. Further, since we are performing this estimation in order to make a bid on a specific, given task, we are interested in predicting the future surplus at this host in the interval X, where X is: estimated arrival time should this task be awarded here minus the deadline of the task.

The local factors include all known guaranteed periodic and aperiodic tasks (100% accurate information), estimates of system overhead (most of

this is known since most system tasks run as periodic tasks), and an estimate of new local arrivals which will be guaranteed and "probably" run within the interval X. This latter estimate is relatively simple if one is confident about the distribution and average arrival rate of tasks.

The factors to consider relating to system-wide load balancing include all the information concerning outstanding bids made by this host. This includes the number of bids, when each was made, the characteristics of the tasks bid on (estimated arrival time, computation time and deadline), and some method of estimating the likelihood of acceptance of a bid. This latter estimation problem can be approached in many ways. It is our opinion, however, that simply keeping track of percentage of bids accepted in the past is not a good idea and may produce the same ineffective results as when we used the moving average forecast technique. The real-time environment gives rise to too many different, very precise conditions to make keeping overall averages very suitable. It may be better for each host to use (out-of-date) state information about the loads of other hosts to estimate the likelihood of a task being awarded to it. Then, in calculating estimated surplus, each outstanding bid made by a host is treated as if it will definitely be awarded or not depending on the likelihood associated with it.

2.2 Bidding in Soft Real-Time Systems (By Gary Rommel)

The research performed in this quarter was in three main areas:

1. further development of our simulation test-bed,
2. development of queuing models of a distributed system, and
3. the development of a proposed general network model.

We will now briefly describe each effort.

2.2.1 Test-Bed

First, we have been able to enhance a general purpose test-bed simulation program. This program allows us to test clusters and distributed groups under various scheduling algorithms.

The unique feature of a process in our model is that the service rate of the process is a function of the members of the group at the process's site as well as a function of the host performance.

In the case of a cluster, the service time of a process at a particular site is given by the following

$$E(s) = E(so)*\{1 + x*(1 - n/nmax)\}$$

Where

E(s) - service time of a process of the cluster

E(s0) - service time of a process of the cluster if all members were present at one host. This parameter is host dependent.

n-number of members present at a given host

nmax-number of members in a cluster

x-factor determining interprocess overhead for remote site processes.

The cluster might be used to model communication bound processes. We might expect to see a cluster in a real time environment when one process reads an A/D converter, another process performs some analysis on the data, and still another stores the data. These processes may be on different hosts, but performance would be improved if they were on the same host.

For a distributed group the process service time is determined by the following

$$E(s) = E(s0)*\{1 + (x/(nmax - 1))*(n - i)\}$$

Distributed groups model processes which communicate infrequently or with small amounts of data. We may also use distributed groups to model real-time processes in which communication between processes are limited while computation times dominate.

In our model, we consider process scheduling. Scheduling is based on bidding and the use of the McCulloch-Pitts neuron. The McCulloch-Pitts neuron includes process factors. Among these are process size, remaining time to complete, priority, and unique resource requirements. Likewise, the neuron considers host related states such as queue length, processor speed, current free memory, outstanding bids and host figure of merit. The neuron forms a single value based on these inputs to determine the ability of a process to run at a given host. We have been extending our current model to include actual network delays in our bids, by modeling cluster and distributed group service times, by consideration of outstanding bids, by passing of state data, and the use of a host figure of merit. We have also added the feature that each individual network link has a delay which is a function of the traffic across it. We also reduce the bids by a factor that is a function of the current delay of the bid message through the network.

This value is determined by the time between the request for bid and the reception of the bid.

The figure of merit is a single number relating directly to the response times of the processes at a given host. We are going to study a simple exponential algorithm to predict future response times coupled with the figure of merit. We use two algorithms for the figure of merit: The first is a strictly a static exponential predicator:

$$f(n) = b*f(n - 1) + c*r$$

where

b and c are weighting factors; f(n) is the predicted figure of merit; f(n-1) is the past figure of merit; r is the response time of the process just completed.

The second incorporates the memory size:

$$f(n) = d*b*f(n-1) + (1-d) *c*r$$

where

b and c are weighting factors; f(n) is the predicted figure of merit; f(n-1) is the past figure of merit; r is the response time of the process just completed; d is a factor relating to the current memory size

The memory size factor is given by the following:

$$d = (mmax-mfree)/mmax$$

where

mmax is the maximum memory at the site,

mfree is the free memory at the site.

By using this function, the host figure of merit is more dependent on process response times when memory is free than not. We choose this because

we expect that when memory is free the last response will predict the next since no queueing need be considered.

The figure of merit is also reduced as a function of time and free memory. This is done using a daemon process called at a given period.

$$f(n) = f(n-1) * d * e$$

where

f is the figure of merit

d is the memory factor

e is the tuning factor.

2.2.2 Distributed Networks and Queuing Analysis

Our second effort in soft real time scheduling is in the area of queuing techniques. We are interested in using queueing analysis to extend our own simulation models. Once our algorithm is developed, we will simulate the algorithm to verify it. However, due to the cost of simulations we would like to extend our results by queueing analysis. Here we have developed models of distributed networks in which the scheduling algorithms are strictly queue length dependent.

The major effort in this research is the use of optimization based on operational research techniques. Our models here use Kleinrock's notion of independence to model the effect of the network delay. Current effort is to use either the gradient or complex conjunction techniques to determine the optimal flows.

2.3 Defense of Thesis

Ravi Mirchandaney successfully defended his thesis (supported by this grant) entitled, "Decentralized Job Scheduling Using a Network of Stochastic Learning Automata." The major results of this thesis have been reported in previous quarterly reports. Rewriting, preparation of the defense, and the actual defense occurred during this quarter. We are now working on generating a paper from the thesis to submit to a journal.

3.0 Conclusions and Recommendations

Stability in distributed scheduling algorithms is a very difficult problem. Stability issues are specific to the algorithm, to the environment, and to the designer's subjective ideas as to what constitutes reasonable behavior. There is no clear cut definition for stability, unless one is willing to define it in very broad terms. But, even in that case, the developer of the algorithm needs to understand how to control the algorithm and environment so as to conform to the definition. We hypothesize that distributed scheduling algorithms need many special cases to be able to react well under the many diverse system conditions found in distributed systems.

In this quarter we directly addressed and briefly evaluated several specific stability issues of two algorithms: one based on stochastic learning automata and the other based on bidding. We suggest that much more work is required to fully understand, more formally define, and completely evaluate these and other stability issues of distributed scheduling algorithms.

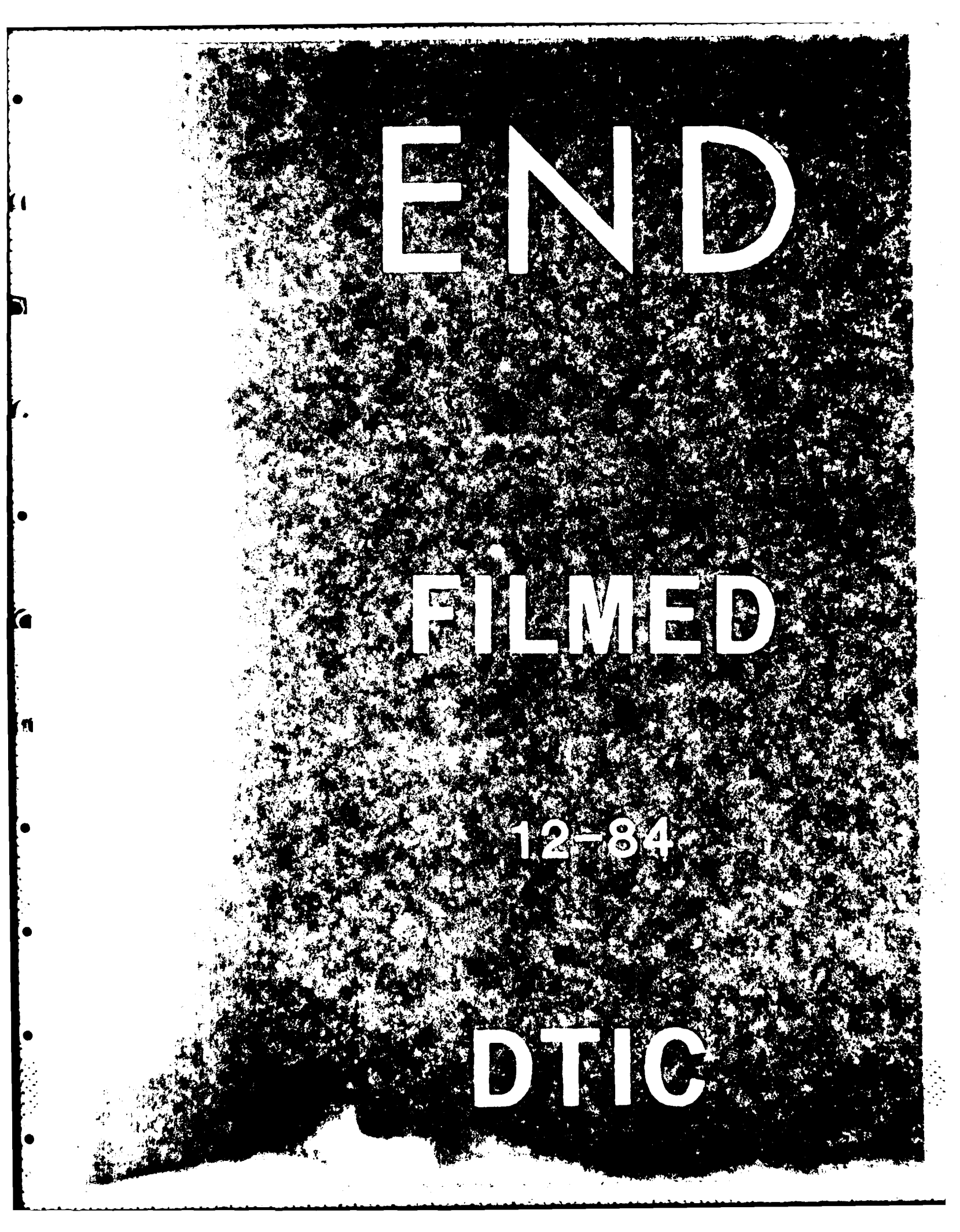
4.0 Budget

Spent prior to the months listed below: $66,464.62.

Month	Planned	Actually Spent
December (83)	$ 3,319.21	$ 3,319.21
January (84)	2,336.68	2,336.68
February	7,323,58	7,323.58
March	5,376.75	5,376.75
April	-513.36	-513.36
May	4,000.00	
June	8,000.00	8,753.47 (for 2 mos.)
July	8,000.00	5,786.42
August	7,000.00	
September	5,000.00	
October	5,000.00	
November	5,000.00	
December	4,522.52	
Total Planned and Spent	$130,830.50	98,847.37
Balance		$ 31,982.63

5.0 Distribution List

Defense Technical Information Center Cameron Station Alexandria, VA 22314	2 copies
Commander USAERADCOM ATTN: DELSD-L-S Fort Monmouth, NJ 07703	2 copies
Commander USACECOM ATTN: DRSEL-COM-RF (Dr. Klein) Fort Monmouth, NJ 07703	1 copy
Commander USACECOM ATTN: DRSEL-COM-D (E. Famolari) Fort Monmouth, NJ 07703	1 copy
Commander USACECOM ATTN: DRSEL-COM-RF-2 (C. Graff) Fort Monmouth, NJ 07703	10 copies


END
FILMED
12-84
DTIC