AN EFFICIENT PROTOCOL FOR LOAD BALANCING ON CSMA/CD NETWORKS

Benjamin W. Wah and Jie-Yong Juang
School of Electrical Engineering
Purdue University
W. Lafayette, IN 47907

ABSTRACT

In this paper, we have studied the load balancing of jobs on CSMA/CD networks. Due to the broadcast capabilities of these networks, workload status can be distributed easily to all the processors. Since only one job can be sent over the network at any time, it is essential that a job from the processor with maximum response time be sent to the processor with minimum response time. To support load balancing, we have proposed a unified window protocol for resolving contention and priorities on the system, and finding the processors with the maximum and minimum load. The scheme is reduced to the general problem of finding the minimum from a set of random numbers. Analytical and simulation results show that the average number of contention slots is a constant and approaches e. The scheme is also applicable to the general problem of resolving contention on CSMA/CD networks.

KEYWORDS AND PHRASES: CSMA/CD network, contention, load balancing, priority, response time, window protocol.

1. INTRODUCTION

The recent advances in large-scale integrated logic and communication technology, coupled with the explosion in size and complexity of new applications, have led to the development of parallel and distributed processing systems. These systems may possess a large number of general and special purpose processing units interconnected by a network.

One of the functions of the network is to allow the sharing of resources that include the access of programs and databases, and the sharing of computational facilities. *Load balancing* is a scheme which engages communication facilities in supporting remote job execution in a user-transparent manner so that the utilization of resources is improved through the enhancement of resources sharing. Depending on the workload of processors on the network, the network operating system may distribute newly arrived jobs to a remote processor, or may schedule them for local execution. The concept of load balancing has been implemented in the Engineering Computer Network at Purdue University [HWA82]. Operational experiences with the system reveal that load balancing has contributed significantly to the improved resource utilization and reduced response time.

Theoretical studies on load balancing have been done with respect to centralized control [CHO79, NI81]. Newly arrived jobs are routed to different processors by a scheduling policy. Two classes of scheduling strategies have been investigated, namely, probabilistic and deterministic. In a probabilistic routing strategy, an arriving job is dispatched to the processors according to a set of branching probabilities. These probabilities are associated with the statistics gathered about the processing speeds and job arrival history. On the other hand, a deterministic strategy routes jobs to the processors according to the current workload status. This is similar to a probabilistic strategy except that the branching probabilities are updated dynamically whenever the system state changes.

Previous studies on both strategies have shown that a multiprocessor system with load balancing performs much better than one without it, and its performance is nearly as good as that of a single fast processor. However, three problems occur when these strategies are applied to a system connected by a relatively slow network. First, the success of the above strategies lies in the fact that the information on system load are readily available without any delay. When the network is slow, significant delays may be incurred in distributing the status information. The routing of jobs based on inaccurate information may have an adverse effect on performance. Second, the distribution of status information may have a large overhead on the network traffic. Special considerations may have to be made to reduce this overhead based on the network characteristics. Third, load balancing tends to increase network load and impede message traffic. Tradeoffs must be made to balance the delay of jobs and messages.

In this paper, we concentrate on the design of a protocol for load balancing on CSMA/CD type networks [TAN81, MET76]. An organization of the system is shown in Figure 1. These networks are characterized by a single shared bus with a distributed contention resolution protocol. The unit of message is a packet and can be sent in a time slot. Whenever multiple processors try to access the bus, they detect whether a carrier is present (the network is being used). If the bus is busy, they wait until it becomes free. If it is free, they start the transmission after a random delay. When two processors try to transmit simultaneously, a *collision* is said to occur. The contending processors have to transmit again in the future. The special broadcast capability of these networks facilitates the efficient distribution of status information for load balancing. However, the major drawback is the limited bandwidth. Only one job can be distributed at any time over the network. Therefore, it is important that jobs from the processor with the max-

This research was partially supported by National Science Foundation Grant ECS80-16580 and by CIDMAC, a research unit of Purdue University, sponsored by Purdue, Cincinnati Milicron Corporation, Control Data Corporation, Cummins Engine Company, Ransburg Corporation, and TRW.

8th Conference on Local Computer Networks, Minneapolis, Minnesota, October 17-19, 1983.

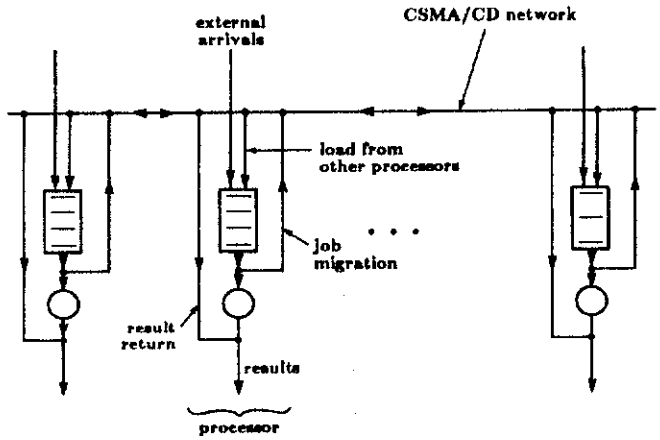

Figure 1. Organization of system connected by a CSMA/CD network

imum load migrate to the processor with the minimum load.

This paper is divided into five sections. A MAX-MIN load balancing strategy is proposed in Section 2. An efficient CSMA/CD protocol which can support load balancing and message transfer is discussed in Section 3. This protocol is an optimal collision detection scheme with an average behavior of e contention slots regardless of the number of contending stations provided that this number can be assessed quite accurately. Although under heavy load, the scheme does not switch to a time-division multiplexed mode as what is done in the urn protocol of Kleinrock and Yemini [KLE78], the urn protocol cannot be adapted for load balancing and prioritized access. Some performance results are shown in Section 4.

2. MAX-MIN LOAD BALANCING STRATEGY

Due to the limitation of CSMA/CD networks, only one job can be sent over the network at any time. It is essential that jobs from processor with the heaviest load migrate to processor with the lightest load. In order to support this MAX-MIN load balancing strategy, the following tasks are involved:

(a) to decide when load balancing is needed;
(b) to identify the processors with the maximum and minimum load;
(c) to send a job from the processor with maximum load to the processor with minimum load;
(d) to send the results of execution back to the originating site.

Migrating jobs and returning results can be handled in the same way as regular message transfer. However, the decision to perform load balancing or not depends on the network traffic. If the total round trip delay of sending jobs over the network and the execution time on the remote processor is longer than the local execution time, then load balancing is not beneficial. Monitors on network traffic must be kept at each processor. When it is decided that load balancing is necessary based on previous status information collected, the workload status of all the processors must be determined in order to identify the source and sink of the job to be balanced. A simple way is to go through all the processors in a round-robin fashion. This is time-consuming because it depends on the number of processors on the network. We propose in the next section an efficient protocol to resolve this problem. As a byproduct, the protocol can also be used for collision detection of messages.

3. A CSMA/CD PROTOCOL FOR SUPPORTING MAX-MIN LOAD BALANCING STRATEGY

There are four types of tasks to be serviced by the protocol. These include regular message transfer, result return, job migration, and MAX-MIN identification. They are listed here in descending order of priorities. Since the network is originally designed for message transfer, we assign the highest priority to messages. Load balancing can be done by using the spare channel capacity. The return of results from a remote processor is important because the delay contributes to the response time of jobs. It is also natural that previous load balancing process should have higher priority than that of a newly invoked one. Thus, MAX-MIN identification is assigned the lowest priority.

The identification of the processor for using the channel consists of two phases. First, the group of processors with the highest priority must be determined. After these processors are known, a contention phase follows which identify a unique processor to transmit. These phases must be carried out for every packet sent in the system except when messages are sent. In this case, only the contention phase is necessary because the priority class remains at the highest level until all the messages are sent. A summary of the protocol is shown in Figure 2.

The effect of our priority assignment favors the transfer of messages. However, it would reduce the possibility of load balancing in the system. Suppose the maximum (resp. minimum) response time in the system is r_{max} (resp. r_{min}) and the transmission delay of sending a job over the network is s_j. Let the channel load for messages be c_m. Since messages have higher priority, the average total delay of sending the job is $s_j/(1-c_m)$. Assuming that the return of results takes the same time, load balancing will be effective if,

$$r_{max} > r_{min} + \frac{2\,s_j}{1 - c_m}$$

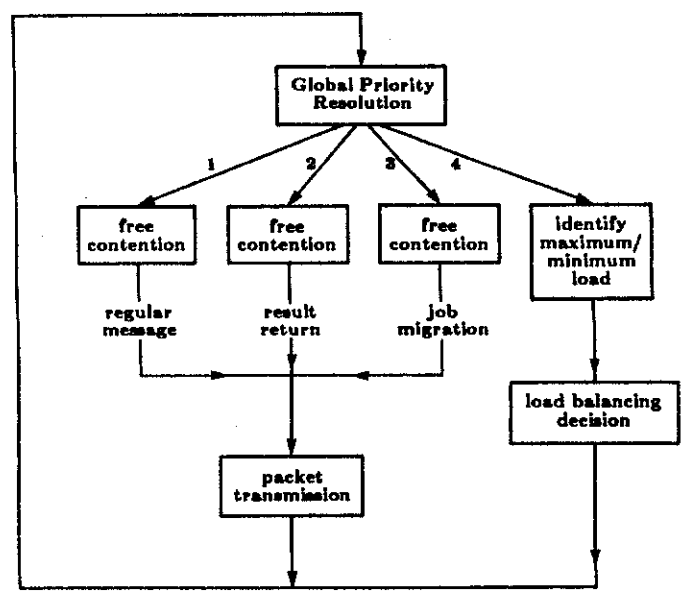

Figure 2. A CSMA/CD protocol supporting MAX-MIN load balancing strategy

or

$$c_m < 1 - \frac{2\,s_i}{r_{max} - r_{min}} \qquad (1)$$

That is, although the channel is idle, load balancing cannot be carried out because the channel delay is too high. A way to improve the channel delay for load balancing is to reduce the priority of messages. This reduces the delay of job and result transfer at the expense of message delay.

3.1. The Resolution of Priorities

Several prioritized CSMA protocols have been suggested in recent years [TOB82, SHA83]. Tobagi proposed that priorities be resolved linearly [TOB82]. In this case, each processor is assigned the highest priority of the local messages. During the resolution of priorities, a slot is reserved for each priority class. A processor will transmit during the slot reserved for the priority. The resolution scheme will stop when the highest priority level is determined. This scheme is good when high priority messages are predominantly sent.

A closer look at the problem shows that the priority resolution problem searches for the highest priority class of the processors. This is similar to finding the processor with the maximum load in the MAX-MIN strategy except that in this case, the priority levels are concerned. We will show an efficient scheme for finding the maximum (resp. minimum) in the next section.

3.2. The Resolution of Contention

We present in this section a unified scheme for finding the minimum of a set of numbers. (Finding the maximum can be transformed to finding the minimum using an inverse function.) This is the basic operation performed in the resolution of contention and priorities.

(a) Contention resolution for packet transmission: When multiple processors wish to transmit, each processor generates a random number in the interval (0,1]. The processor assigned the channel is the one with the minimum number.

(b) Identification of processor with maximum (resp. minimum) load: The load on a processor is characterized by the response time of executing a job on that processor. The response times of all the processors are distributed in the interval $[0,\infty)$. To find the processor with the maximum (resp. minimum) load is equivalent to finding the maximum (resp. minimum) of a set of random numbers in the interval $[0,\infty)$.

(c) Identification of processors with maximum priority: The priority of a processor is an integer from the set $\{0, 1, ..., p_{max}\}$. The identification problem searches for the maximum priority value of all the processors in this interval.

These problems can be generalized as follows. Given a set of random numbers y_i, $i = 1, 2, ..., n$, in the interval (0,1] with distribution F(y) such that $y_1 \le y_2 \le \cdots \le y_n$, the problem is to identify the processor(s) with value y_1. Transformation may have to be performed on the original set of random numbers in order to result in the required distribution. It should be noted that in many cases, only an estimate on n can be obtained.

3.3 Optimal Window Protocol

A general method to solve the search problem is to use a window protocol that is outlined in Figure 3. Given that the minimum value is sought, a first estimate on the window can be made. This value may be

```
procedure search (n̂, y1, y2, ..., yn̂, window);
{
/* n̂ - number of elements,
   y1, ...,yn̂ - set of random numbers of distribution F(y),
   window - function to calculate window size w,
   lb_window - lower bound of window to be searched,
   ub_window - upper bound of window to be searched.
*/

lb_window = 0;
ub_window = 1;
w = window( lb_window, ub_window, n̂);

while (TRUE) do {
     if (y1 ≥ w and y2 ≥ w) then
          lb_window = w;    /* update lb */
     else
          if (y1 < w and y2 < w) then
               ub_window = w;    /* update ub */
          else {
               transmit_packet;  /* successful trans. */
               return;
               }
     w = window( lb_window, ub_window, n̂);
     }
}
```

Figure 3. Contention resolution protocol for searching y_1.

estimated from prior value of y_1 found and the distribution F(y). Let this value be w. Assuming that the lower bound of the interval to be searched is 0, and the upper bound is 1 initially, only processors with values in the range (0,w] are allowed to transmit in the first contention slot. If exactly one processor transmits, the minimum value is found. If collision occurs, the minimum must lie in the interval (0,w], and the upper bound is updated to w. Otherwise, there is no transmission, and the minimum must lie in the interval (w,1]. The lower bound is updated to w accordingly. The last case is equivalent to saying that collision would occur if the interval (w,1] is used.

Unless the transmission is successful, we can always identify an interval (a,b] such that at least two of the y_i's lie in (a,b] and no y_i is smaller than a. This condition is designated as event A.

A = { at least two y_i's are in (a,b] given that all y_i's are in (a,1]}

In order to isolate the processor with the minimum y_i, the size of the window has to be shrinked. Let the reduced window be (a,w] such that $a < w < b$. Transmission will be successful if there is only one of the y_i's falling in (a,w]. This event is denoted as B.

B = { exactly one of the y_i's is in (a,w], all others are in (w,1]}

Thus the probability that exactly one station transmits in the next contention slot is $g(w) = Pr\{B|A\}$. We derive the optimal value of w so that g(w) is maximized. Since

$$Pr\{B|A\} = Pr\{A,B\}/P\{A\},$$

$$Pr\{A\} = [1-F(a)]^n - [1-F(b)]^n - n[F(b)-F(a)][1-F(b)]^{n-1},$$

and

$$\Pr\{A,B\} = n[F(w)-F(a)]\{[1-F(w)]^{n-1}-[1-F(b)]^{n-1}\}.$$

Thus

$$g(w) = \Pr\{A,B\}/P\{A\} = K[F(w)-F(a)]\{[1-F(w)]^{n-1}-[1-F(b)]^{n-1}\} \quad (2)$$

where $K = n/P\{A\}$.

In order to find an optimal value of w, we solve for w in $\frac{d}{dw}[g(w)] = 0$, and this leads to Eq. (3) below with the assumption that $f(w) \neq 0$.

$$[1-F(w)]^{n-1} - [1-F(b)]^{n-1} - (n-1)[F(w)-F(a)][1-F(w)]^{n-2} = 0 \quad (3)$$

Let $x = 1-F(w)$, Eq. (3) becomes

$$x^{n-1} - \frac{(n-1)[1-F(a)]x^{n-2}}{n} - \frac{[1-F(b)]^{n-1}}{n} = 0 \quad (4)$$

It can be shown that a real root of Eq. (4) exists and satisfies $(1-F(b)) < x_o < (1-F(a))$. The optimal window w_o can be computed directly from x_o as in Eq. (5).

$$w_o = F^{-1}(1-x_o) \quad (5)$$

There is no close form solution to Eq. (4). A numerical method can be applied to obtain x_o. Nevertheless, it may not be fast enough for practical application. An approximate solution is derived here. Rewriting Eq. (2) into the form of Eq. (6), we have,

$$g(w) = K[F(w)-F(a)][F(b)-F(w)][1-F(w)]^{n-2}\sum_{i=0}^{n-2} z^i \quad (6)$$

where $z = [1-F(b)]/[1-F(w)]$. An approximation function $\hat{g}(w)$ is given by substituting the term $\left[\sum_{i=0}^{n-2} z^i\right]$ by (n-1) since z is very close to 1. That is,

$$\hat{g}(w) = K'[F(w)-F(a)][F(b)-F(w)][1-F(w)]^{n-2} \quad (7)$$

where $K' = (n-1)K$. $\hat{g}(w)$ can be maximized by solving $\frac{d}{dw}[\log \hat{g}(w)] = 0$. From this, we obtain Eq. (8),

$$\frac{f(w)}{F(w)-F(a)} + \frac{f(w)}{F(w)-F(b)} + \frac{(n-2)f(w)}{F(w)-1} = 0 \quad (8)$$

or equivalently,

$$[F(w)]^2 + C[F(w)] + D = 0, \quad (9)$$

where

$$C = -\frac{(n-1)[F(a)+F(b)]+2}{n}$$

$$D = \frac{F(a)+F(b)+(n-2)[F(a)][F(b)]}{n}.$$

A solution of Eq. (9) falling in the interval (F(a),F(b)) is given by

$$F(w_a) = \frac{-C-\sqrt{C^2-4D}}{2} \quad (10)$$

The approximate optimal window, w_a, can then calculated from the inverse distribution function.

We will show empirically that the approximate window control rule performs nearly as good as the optimal one, while its computational efficiency makes it more preferable to the optimal control rule.

Both the optimal and approximate control rules rely on the knowledge of the distribution of random numbers, F(y), and the number of contending processors on the network, n. The former can be obtained either analytically or empirically. A future paper shows an iterative method for obtaining the distribution function. The latter parameter has to be estimated dynamically. This

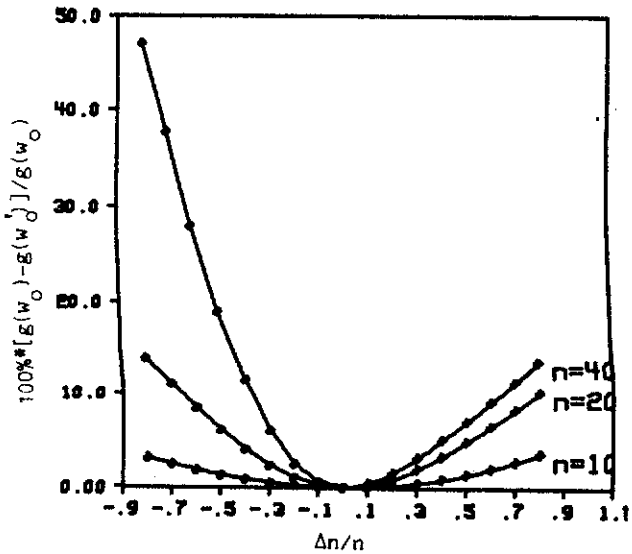

Figure 4. Sensitivity of success probability with respect to load estimation error (window = (0.3, 0.4]; w_o = optimal value of w; w'_o = optimal value of w_o with load estimation error).

can be estimated based on the previous value of y_1. Since the mean of y_1 is a function of n, a simple estimate of n can be obtained by assuming that the number of contending stations does not change, and the previous value of y_1 exists at its mean. However, an accurate estimation may be difficult in practice. Fortunately, the scheme can tolerate a high degree of estimation error. In Figure 4, the sensitivity of success probability is plotted against the load estimation error. It shows that the degradation can be as small as 2% if the accuracy of estimation is within 20%.

When n=2, both Eq.'s (4) and (9) have the same solution, i.e. $F(w_o) = [F(a)+F(b)]/2$. If F(y) is uniformly distributed over (0,1], then $w_o = (a+b)/2$. This binary dividing control rule can be used as a heuristic for general distribution functions. It can be interpreted as one which always predicts that there are two contending processors. As a result, it performs satisfiably when channel is lightly loaded but degrades drastically when the channel load is heavy.

The dynamic expansion and shrinking of window corresponds closely to the urn protocol of Kleinrock and Yemini [KLE78], and the time window protocol of Kurose and Schwartz [KUR83]. In the urn protocol, the set of random numbers are the spatial identifiers of the processors. In the time window protocol, the arrival times of messages at processors are used. Our proposed method here represents a unified approach. We have used a general distribution in developing the optimal rule. Further, if the set of random numbers are regenerated after each successful transmission, no distinction is made between old and new contending stations. This preserves the fairness to all the processors.

4. PERFORMANCE EVALUATION

The performance evaluation of the protocol consists of two parts, the evaluation of the contention resolution protocol, and the evaluation of the MAX-MIN load balancing strategy.

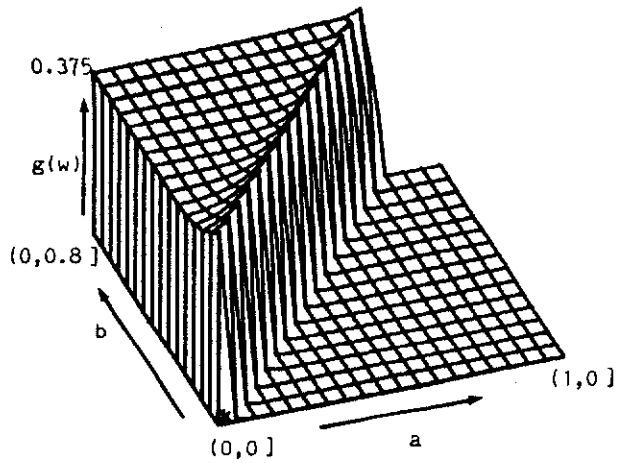

Figure 5. Optimal success probability of contention with respect to window (a,b] (number of contending processors = 20).

The throughput of a CSMA channel can be derived from the average number of contending slots experienced before a packet is transmitted. The probability of success in the i-th contention slot is $g_i(w)$ as given by Eq. (4). The average number of contention slots is,

$$\bar{c} = \sum_{i=1}^{\infty} i\, g_i(w) \prod_{j=1}^{i-1} [1-g_j(w)] \qquad (11)$$

$g_i(w)$'s are functions of the window size, number of contending processors, and window control rule. We have evaluated g(w) with respect to different window sizes and number of contending processors under the assumptions that the distribution is uniform in (0,1], and the number of contending stations is assessed accurately. In Figure 5, g(w) is plotted against various windows for the optimal control strategy with 20 contending processors. It can be seen that the values of g(w) are almost constant except for those of small windows, that is, when a is close to b. When the window size is small, it is more probable that a smaller number of y_i's lie in this interval, and the uncertainity about the position of y_1 decreases. Therefore, g(w) increases as the window becomes smaller. In Figure 6, g(w) is plotted against the number of contending processors for various window sizes under the optimal control rule. As the number of contending processors increases, g(w) decreases and approaches 0.375. Replacing the $g_i(w)$'s in Eq. 10 by this value, we obtain $\bar{c} = 2.67 = e$.

By using the approximate control rule, the degradation in g(w) is very small. This is depicted in Figure 7 for various window sizes. We observe that the error increases as the window size is decreased. This is due to the fact that a deviation in the value of w results in a larger relative error when the window size is small. However, the error never exceed 2% of the optimal value. In Figure 8, the relative error is plotted against different number of contending processors. For a given window size, we see that the relative error peaks at certain number of contending processors. When the number of stations is large, the given window size is relatively large. Therefore, the error is small. On the other hand, when the number of stations is small, the error due to the approximation in Eq. (6) is small. All these results demonstrates that the approximate control rule performs competitively.

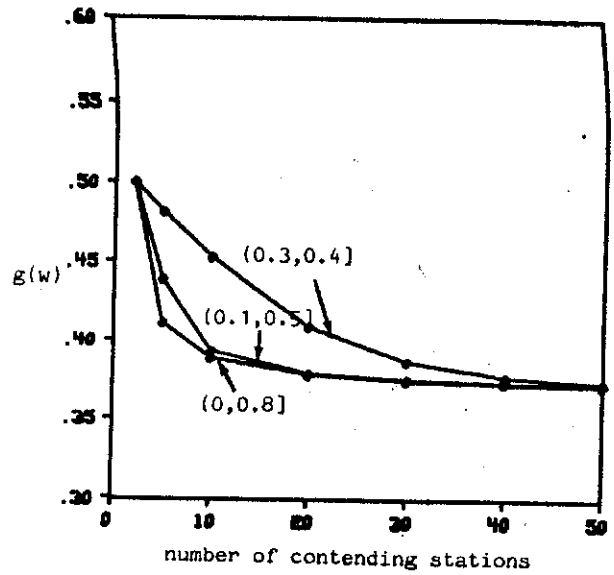

Figure 6. Optimal success probability of contention with respect number of contending processors.

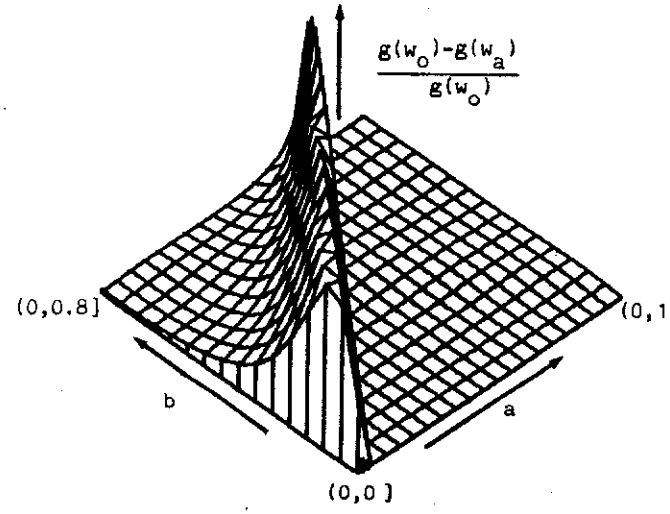

Figure 7. Degradation in success probability for the approximate control rule with respect to window (a,b] (number of contending processors = 20; w_o = optimal value of w; w_a = approximate value of w).

A simulator has been developed to evaluate the performance of the window protocol. These results are plotted in Figure 9. The average number of contention slots is about 2.7 for both the optimal and approximate rules as the number of stations increases. The binary divide rule also performs well when the number of contending processors is small. However, the average number of contention slots is almost doubled when n is large. This phenomenon has been predicted in the last section. The above results were based on the fact that the number of contending stations was know. When there is error in

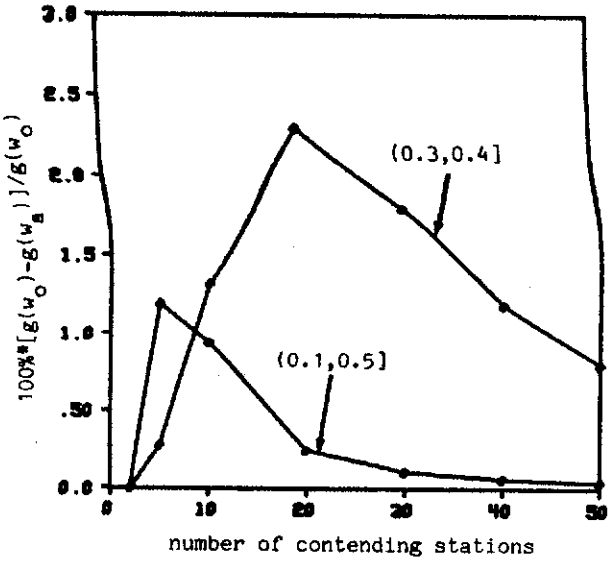

Figure 8. Degradation in success probability for the approximate control rule with respect to number of contending processors.

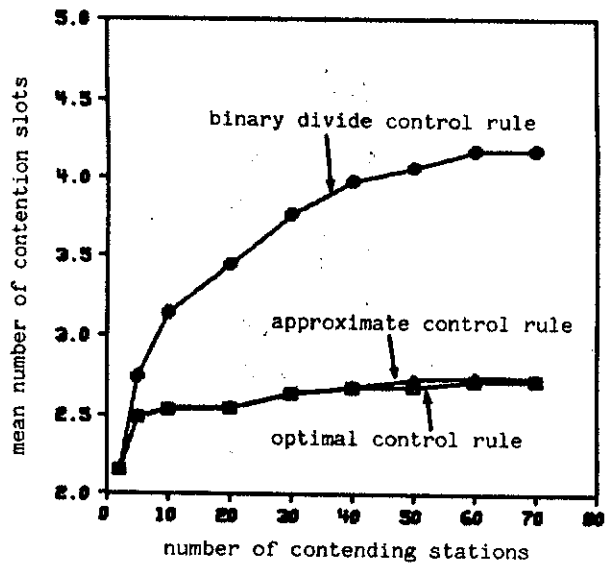

Figure 9. Performance of window protocol operating under different window control rules.

estimating this number, the number of contentions slots will increase.

A simulator has also been developed to evaluate the performance of load balancing supported by the window protocol. The simulator was written in ASPOL and assumed a fixed number of processors in the system. It was assumed that job arrivals are Poisson at each processor and the service time is exponentially distributed. The normalized mean job response time is plotted with respect to different job traffic with no message traffic in Figure 10. The results are compared against the case without load balancing, that is, the case of multiple independent M/M/1 queues. We find that the response time is always better when load balancing is applied.

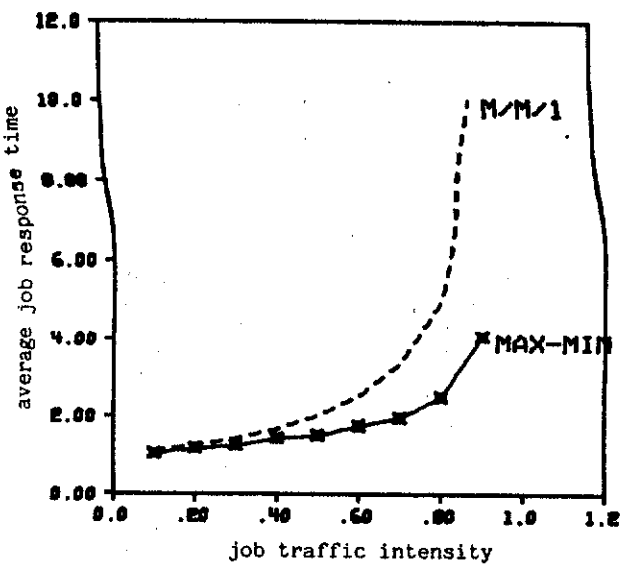

Figure 10. Reduction in response time under load balancing with no message traffic (number of processors = 20; ratio of service to transmission delays = 1.0).

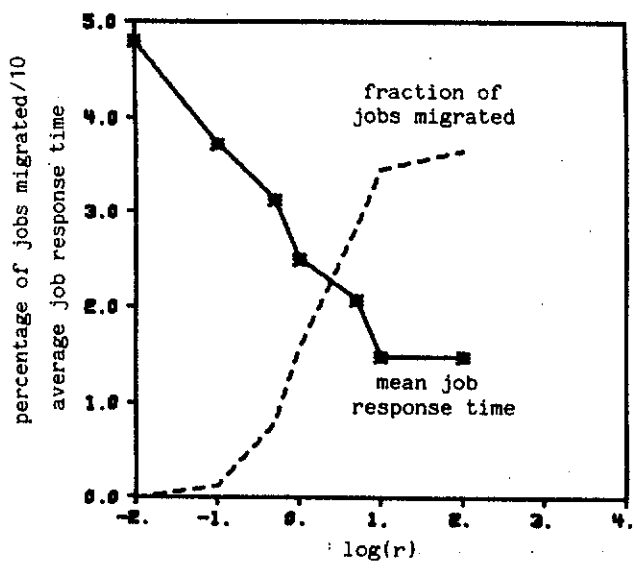

Figure 11. Effect of ratio of service to transmission delays, r, on load balancing.

Since almost no queue is developed at each processor when the job traffic is low, load balancing is less effective. In Figure 11, we show the job response time with respect to the ratio of job service time and transmission delay. The percentage of jobs being migrated is also shown. As we can interpret from this figure, fast processors coupled with a slow communication channel (r is small) will not be benefited from load balancing. As the speed of communication channel increases, more jobs are migrated and the response time is improved. However, when the channel speed exceeds certain threshold, load balancing can provide no further improvement. In Figure 12, the effects on load balancing due to message traffic is shown. As the

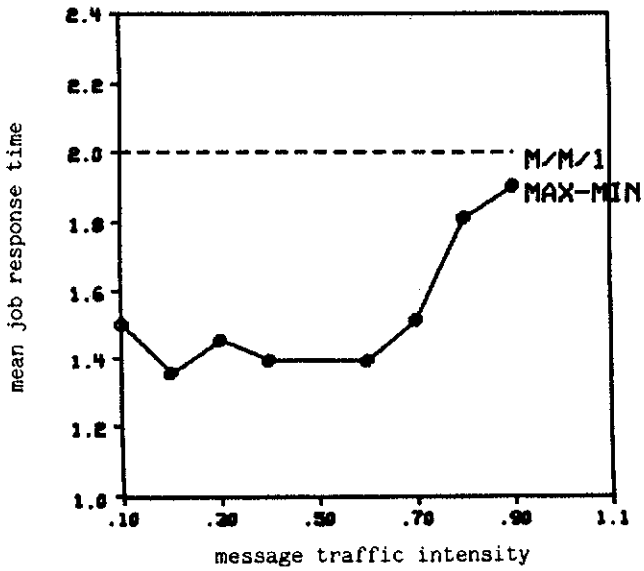

Figure 12. Effects on load balancing due to message traffic (job traffic intensity at each processor = 0.5; number of processors = 20).

message traffic becomes higher, the effective channel capacity is less and load balancing is not carried out as often.

5. CONCLUSION

In this paper, we have proposed a load balancing strategy on CSMA/CD networks. Due to the broadcast capabilities of these networks, system status can be distributed easily to all the processors. However, only one job can be sent at any time over the network. We study the MAX-MIN strategy which sends a job from the processor with the maximum response time to the processor with the minimum response time. To support the scheme on CSMA/CD networks, three different tasks are identified, namely, resolving contention, resolving priorities, and finding the maximum and minimum response times. We proposed a unified model that reduces these problems to finding the minimum of a set of random numbers. A window protocol is proposed to identify the minimum. Optimal and approximate control rules for the window are evaluated. It is found that under uniform distribution, the average number of contention slots is approximately e regardless of the number of contending processors provided that this number can be assessed quite accurately. One important application of the window protocol is in the resolution of contention of message transmission.

REFERENCES

[CHO79] Chow, Y.C., and W. Kohler, "Models for Dynamic Load Balancing in a Heterogeneous Multiple Processor System," *IEEE Transactions on Computers*, Vol. C-28, No. 5, May 1979, pp. 334-361.

[FRA80] Franta, W. R. and M. B. Bilodeau, "Analysis of a Prioritized CSMA Protocol Based on Staggered Delays," *Acta Informatica*, 13, 1980, pp. 299-324.

[HWA81] Hwang, K. et al., "A Unix-Based Local Computer Network With Load Balancing," *IEEE Computer*, April 1982, Vol. 15, No. 4, pp. 55-66.

[KLE78] Kleinrock, L., and Y. Yemini, "An Optimal Adaptive Scheme for Multiple Access Broadcast Communication," *Proc. ICC*, pp. 7.2.1-7.2.5, 1978.

[KUR83] Kurose, J. F., and M. Schwartz, "A Family of Window Protocols for Time Constrained Applications in CSMA Networks," *Proc. IEEE INFOCOM 83*, San Diego, CA, 1983, pp. 405-413.

[MET76] Metcalfe, R.M., and Boggs, D.R., "Ethernet : Distributed Packet Switching for Local Computer Networks," *CACM*, Vol.19, July 1976, pp.395-404.

[NI81] Ni, L.M. and Hwang, K. "Optimal Load Balancing Strategies for a Multiple Processor System," *Proc. of 10th Int'l Conf. on Parallel Processing*, Aug. 1981, pp. 352-357.

[SHA83] Shacham, N., "A Protocol for Preferred Access in Packet-Switching Radio Networks," *IEEE Trans. on Comm.*, Vol. COM-31, No. 2, Feb. 1983, pp. 253-264.

[TAN81] Tanenbaum, A. S., *Computer Networks*, Prentice Hall, Inc., New Jersey, 1981.

[TOB82] Tobagi, F. A., "Carrier Sense Multiple Access with Message-Based Priority Functions," *IEEE Transactions on Comm.*, Vol. COM-30, No. 1, January 1982.