CHANNEL ALLOCATION IN MULTIPLE CONTENTION BUS NETWORKS

Jie-Yong Juang and Benjamin W. Wah

ABSTRACT

Channels are allocated to packets of high priority in a priority-based channel-allocation strategy. The identification of packets of the highest priority in a network is equivalent to determining the minimum among a set of distributed random numbers. In this paper, a multi-window protocol is proposed for priority-based channel allocation in a network with multiple CSMA/CD channels. The strategy partitions the domain of priority levels into intervals, and utilizes the collision-detection capability of contention busses to resolve their status. The average number of contention steps to identify t packets of the highest priority out of N packets is about $0.8 \cdot \log_2 t + 0.2 \cdot \log_2 N + 1.2$. A degenerate version of the proposed protocol that works on a single bus can be adapted to estimate channel load and is essential for the implementation of state-dependent routing.

INDEX TERMS: Channel allocation, collision detection, multiple busses, ordered selection, priority, state-dependent routing.

1. INTRODUCTION

In this paper, the problem of allocating multiple CSMA/CD channels to processors that generate bursty traffic is studied. Both priority-based and non-priority-based channel-allocation disciplines are discussed. In allocating channels without priorities, the channels are randomly allocated to processors; whereas in a priority-based allocation, each packet is attributed with a priority level, and channels are allocated to packets of the highest priority.

Priority-based CSMA/CD protocols for single contention-bus network have been extensively studied before [Tob82, GoF83, NiL83, Sha83, WaJ85]. These protocols rely primarily on information fed back from the channel to eliminate transmission requests of lower priority and reduce further contention. In *linear priority-resolution protocols* [Tob82, GoF83], requests of higher priority are assigned a shorter delay to access the channel than those of lower priority. Requests of lower priority give up further contention when a transmission is detected indicating the presence of one or more requests of higher priority. In a tree-based priority-resolution scheme [Nil83, Sha83], requests are ordered according to priorities in the terminal nodes of a binary tree. The existence of requests in a subtree is determined from the outcome of collision detections. It is very difficult to adapt these protocols to resolve global priorities in a network with multiple CSMA/CD channels because the correlation among multiple feedback streams have to be considered. In this paper, a multi-window protocol is proposed to support priority-based channel allocation.

An M-CSMA/CD protocol was proposed by Marsan and Roffinella [MaR82] for non-priority allocation in multiple contention busses. In this strategy, a processor probabilistically determines a channel to transmit. However, many processors may happen to request for the same bus to transmit and leave many other channels idle. To cope with this problem, load balancing schemes developed for multiprocessor scheduling [ChK79, NiH85, Tow80] can be applied. These schemes are useful for probabilistic routing but cannot be applied to state-dependent routing since the number of processors contending for a channel is unknown. An efficient algorithm for state-dependent routing in multiple-contention-bus networks is presented in this paper.

In Section 2, we present an interval-resolution scheme that utilizes the collision-detection capability of CSMA/CD busses to resolve the status of intervals and identify packets of the highest priority. An implementation of this scheme on multiple contention busses, called *multi-window protocol*, is proposed in Section 3. Correctness and performance of this protocol are discussed in Section 4. A degenerate version of this protocol is applied in Section 5 to estimate the channel load in non-priority allocations with load-dependent routing.

2. DISTRIBUTED CHANNEL ALLOCATION

The priority level of a packet is a random variable with a distribution that is site dependent and a function of complicated interactions among tasks executing in the network. For mathematical tractability, the priorities of packets at different sites are assumed to be identically and independently distributed. Hence the priorities of packets currently in the network are a collection of observations drawn from a common distribution. Assume that there are N packets to be transmitted with priorities $x_1 \leq x_2 \leq \ldots \leq x_N$. If t channels are allocated according to priorities, then packets with priorities $x_N, \ldots, x_{N-t+1}$ will be transmitted. Therefore, the problem of channel allocation with priority can be seen as a selection problem to determine the t largest numbers from a set of distributed random variates.

One method to identify the extrema of a set of distributed random numbers is to collect them to a central site before sorting them. This approach is inadequate in resolving priorities since collecting the priority levels would involve a large number of packet transmissions. An ordered selection algorithm based on interval resolution is proposed in this section. The algorithm can be implemented on networks with multiple contention busses without explicit messages.

In the proposed algorithm, the priority levels $x_1, \ldots, x_N$ are first transformed into another set of uniformly distributed random numbers $y_1, \ldots, y_N$ such that $y_j = P_{max} - x_j + \delta$, where P_{max} is the maximum priority level. δ is a small random number that

This work was partially supported by CIDMAC, a research unit of Purdue University, sponsored by Purdue, Cincinnati Milicron Corporation, Control Data Corporation, Cummins Engine Company, Ransburg Corporation and TRW, and the National Science Foundation Grant DMC 85-19649.

J. Y. Juang is with the Department of Electrical Engineering and Computer Science, Northwestern University, Evanston, IL 60201.

B. W. Wah is with the Department of Electrical Engineering and Computer Engineering and the Coordinated Science Laboratory, University of Illinois at Urbana-Champaign, Urbana, IL 61801.

IEEE INFOCOM 86 Conference.

could be site dependent, and is used to break ties in priority levels but does not change the relative ordering of packets, i.e., $y_1 > y_2 > ... > y_N$. The allocation of channels is thus reduced to the determination of the minima from the set of distributed random numbers.

2.1. Distributed Ordered Selection on Multiple Contention Busses

An interval-resolution scheme for priority resolution is described here. It is a recursive scheme that partitions and tests the domain of random variates. An interval is resolved if it is empty or contains exactly one number (i.e., a *success*). An unresolved interval is partitioned recursively until it is resolved. To test whether an interval is resolved or not, a resolution scheme with binary questions and ternary responses is used. A processor is asked whether it generates a number in the interval [a,b], and it will answer "yes" or "no" without further description. The same question is directed to all processors, and the aggregated response is of the ternary type, i.e., there is none, one, or more than one processor that responded positively. Such a question-answering session is isomorphic to the collision detection of a CSMA/CD network. In such a network, a processor is either transmitting or not transmitting during a contention slot. A transmission is equivalent to answering "yes," while no transmission is equivalent to answering "no." The capability of a collision-detection mechanism to detect whether there is none, one, or more than one processor transmitting is equivalent to obtaining the ternary response from the processors.

The above analogy suggests that interval resolution can be done by contentions on a bus. In the proposed algorithm, an interval to be resolved is assigned to a bus. A processor contend to transmit its random number on the bus if its random number falls in this interval. By interpreting the outcome of collision detection, a ternary status of the interval can be determined. For convenience, the interval assigned to a bus is called a *transmission window* of the bus. As the interval-resolution process proceeds, the domain of random variates is partitioned into intervals, each of which is in one of the four possible states: empty, success, collided, or unsearched. The order of a random variate y_j can be determined if all intervals between y_1 and y_j have been resolved. Since verifying the status of an interval is independent of verifying other intervals, multiple intervals can be resolved sequentially or in parallel depending on the number of contention busses available.

The example in Figure 1 illustrates an interval-resolution process. In this example, the order of ten random variates generated by six processors is to be determined, and two contention busses are available. The windows used in each step are labeled w_1 and w_2, and the status each interval after a contention is marked in the figure. After three contention steps, a number generated by Processor 3 can be determined to be the minimum, and the second minimum can also be identified. To determine the order of others, further contentions are necessary.

Processors involved in ordered selections have to know the windows used in each contention. This may be done by assigning a processor to generate the appropriate windows and broadcast them to others. This approach is inadequate since broadcasting incurs a significant overhead. Alternatively, if all processors evaluate an identical algorithm with identical inputs, then the windows can be synchronized without any message transfer.

3. MULTI-WINDOW PROTOCOL FOR ORDERED SELECTIONS

A key issue in an interval-resolution scheme for ordered selection is to determine a proper transmission window for each bus. We have developed a single window-control scheme for the degenerate case of a single contention bus [JuW84, WaJ85]. In this section, we describe a *multi-window control scheme* for parallel interval resolutions on multiple busses.

In the multi-window control scheme, window generations

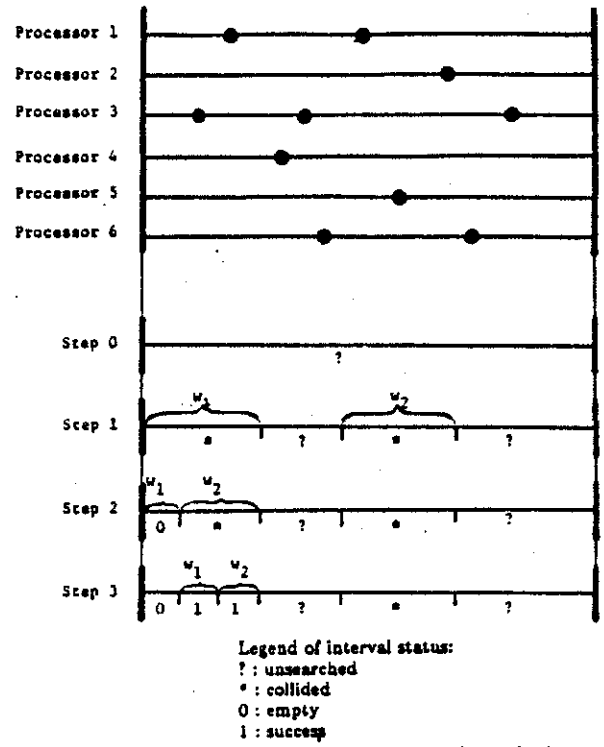

Figure 1. An example illustrating an interval-resolution procedure for ordered selection (windows used in each step are labeled by w_1 and w_2).

in different processors are synchronized by collision detections. A *contention step* consists of the generation of transmission windows, the contention for interval resolution, and the acquisition of interval status by collision detection. Transmission and collision detection can be done in one *contention slot*, which is a fixed system parameter of a CSMA/CD network. On the other hand, generation of transmission windows involves local processing and is subject to optimization.

3.1. Optimal Multi-Window Control

The set of windows used in a contention step is abbreviated as the *window vector* in the sequel. A window vector is chosen from unresolved intervals, including collided and unsearched ones. For convenience, unresolved intervals are represented by vector $\vec{V}$ in which each element is an interval represented by a triplet consisting of the lower and upper bounds and the status of the interval (empty, success, collided, or unsearched). Based on such a representation, the optimization of multi-window control may be formulated in dynamic programming.

Consider the case in which the t smallest numbers are to be selected from N distributed random numbers, and the current unresolved intervals are represented by $\vec{V}$. Given $\vec{V}$, a contention step using contention windows $\vec{w}$ will result in another set of unresolved intervals $\vec{U}$. Denote the expected number of contention steps to complete an ordered selection by $C^t_{N,\vec{V}}$. The dynamic programming formulation for window generation may be expressed recursively as follows.

$$V^t_{N,\vec{V}} = \min_{\vec{w}} \left\{ 1 + \sum_{i=1}^{t} \sum_{\vec{U}} p_{N,i,\vec{V},\vec{U}}(\vec{w}) C^{t-i}_{N-i,\vec{U}} \right\} \qquad (1)$$

where $p_{N,i,\vec{V},\vec{U}}(\vec{w})$ is the probability that in isolating i numbers to be selected with windows $\vec{w}$, the set of unresolved intervals changes from $\vec{V}$ to $\vec{U}$. The optimal window vector is one that minimizes Eq. (1).

To evaluate Eq. (1), all $p_{N,i,\vec{V},\vec{U}}(\vec{w})$'s must be known. Since as many as t busses can be assigned to resolve an interval, there are t^K possible ways of assigning t busses to resolve K intervals. For each assignment, there exists a large number of

combinations of window sizes to be determined. This leads to 3^t possible combinations, each with a different $p_{N,t,\vec{v},\vec{u}}(\vec{w})$. Hence Eq. (1) is too complex to be evaluated, and suboptimal solutions to the window-generation problem will be presented in the next section.

3.2. A Multi-Window Protocol with Heuristic Window Control

(i) *Heuristic Window Control*

Optimization by dynamic programming is intractable due to the large problem space. To tackle the window-generation problem, the following observations were made. First, random variates of the smallest values are to be selected. Thus unresolved intervals at the lower end (that have smaller values) should be searched with higher priority. Second, the proper windows in an interval that maximize the probability of having exact one random variate in each is derived in Appendix A to be u/r if there are r random variates uniformly distributed in an interval of size u. It is also shown that the above interval will most likely contain two random variates if it is a collided one. If two busses are available, the best partition is to divide the domain to be searched into two equal halves. These observations suggest the following heuristic rules.

Rule 1: The entire domain is initially initialized to be an unsearched interval.

Rule 2: An unsearched interval is partitioned into t+1 sub-intervals. The size of the first t sub-intervals is $u/\hat{r}$ each, where u is the size of the unsearched interval, and $\hat{r}$ is the estimated number of random variates in this interval. Note that u is equal to 1, and $\hat{r}$ is equal to N initially.

Rule 3: Unresolved intervals of smaller values (at the lower end) are searched first. Note that there is only one unsearched interval and is always located at the upper end of the domain of values.

Rule 4: After a contention step, a collided interval is partitioned into two equal halves, each of which is considered as an unresolved interval.

Rule 5: Rules (3) and (4) are repeated until either the ordered selection is done, or all the unresolved intervals except the unsearched one at the upper end of the domain have been resolved.

Rule 6: The search is extended into the unsearched interval (by repeating Rule 1) if less than t resolved intervals have been found.

It will be shown in Section 4 that the window control based on these rules is correct and performs satisfactorily.

(ii) *Sequencing and Termination*

If the status of all sub-intervals are known, then the order of the selected numbers and the termination condition of the selection process can be determined. As unresolved intervals are partitioned iteratively, and the number of sub-intervals grows linearly as resolution proceeds, it is hard for a processor to keep track of the status of all sub-intervals because a large amount of memory is needed. It is also not effective when resolution is carried out at a central site.

To reduce the memory requirement, a constant-memory scheme is proposed here. In the proposed scheme, a processor maintains only the approximated order of transmission windows represented as $\vec{Q}$. Each component of $\vec{Q}$ is set to 1 initially, and updated after a contention. The outcomes of collision detection are represented by two t-tuples, $(s_1, s_2, \ldots, s_t)$ and $(d_1, d_2, \ldots, d_t)$, where

$$s_i = \begin{cases} 1 & \text{if contention in the i'th bus succeeds,} \\ 0 & \text{otherwise.} \end{cases} \qquad (2)$$

$$d_i = \begin{cases} 1 & \text{if contention in the i'th bus collides,} \\ 0 & \text{otherwise.} \end{cases}$$

The q_i's are updated as follows.

$$q_i \leftarrow q_i + \sum_{j=1}^{i-1} s_j \qquad i=1, \ldots, t \qquad (3)$$

According to the above updating strategy, q_i is the cumulative number of random variates isolated below the i'th window. It is also the order of the random number in this window if all intervals below it were resolved.

A selection process may be terminated when all numbers to be selected are identified. To know when this is done, each processor maintains an indicator that points to the position where the t'th and (t+1)'th random numbers are most likely to be separated. Assuming that a collided interval contains 2 random numbers, the indicator for termination can be set to the upper end of the k'th window such that the following condition holds.

$$\left| q_k + 2^* \sum_{i=1}^{k} d_i \right| > t \geq \left| q_{k-1} + 2^* \sum_{i=1}^{k-1} d_i \right| \qquad (4)$$

Accordingly, the selection process terminates when all sub-intervals below the termination indicator are resolved. Hence the termination condition may be expressed as

$$\sum_{i=1}^{t} d_i = 0 \quad \text{and} \quad q_t = t \qquad (5)$$

Since the termination indicator is set in such a way that is always larger than the number of elements to be selected, the termination indicator provides an upper bound of the resolution range. This range can be narrowed down as contention proceeds.

The sequencing and termination controls described above will be correct only if all sub-intervals smaller than the termination indicator are searched. We will show in Section 4 that this is true if the proposed heuristic multi-window protocol is used.

The number of contending processors (N) is not known, but can be estimated from the final partition of intervals. We have studied a maximum-likelihood estimation model with auto-regressive moving averages (ARMA) [Jua85, WaJ85] and a maximum-a-priori (MAP) estimate that is more accurate when the underlying random process is known [JuL85]. $\hat{r}$, the number of variates in the unsearched interval can also be estimated using the status information of partitioned intervals. A simple estimation can be obtained by subtracting the estimated number of random variates in the searched range from the total, i.e., setting $\hat{r}$ to be $N - q_t - 2^* \sum_{j=1}^{t} d_j$.

A multi-window protocol based on the above interval-resolution procedure with heuristic window control is outlined in Figures 2 and 3. The protocol is independent of the underlying network except for the function *observe*$(\vec{S}, \vec{D})$. This function is built upon the collision-detection mechanism of a CSMA/CD bus, hence the protocol can be implemented on any multi-bus network with collision-detection capability on every bus.

Since the windows are updated in each step of contention and must be generated before the next contention step begins, a reasonable elapsed time must be set between the completion of collision detection and the initiation of the next contention. To shorten this elapsed time, this protocol must be implemented in hardware. An organization of a hardware implementation is shown in Figure 4a. Detailed design of two time-consuming functions, *transmit* and *window*, are shown in Figures 4b and 4c, respectively. The design of the *transmit* function is simple, hence a transmission decision can be made in less than 100 ns. The *window-generation* function needs arithmetic and logic functions, and is estimated to take several microseconds to complete. It can be shortened if faster device technologies are used.

3.3. Estimation of Channel Load

The multi-window protocol needs to estimate N, the total number of contending packets in the system. Using the proposed window protocol, processors obtain a value w such that the

Jhang & Wah 56 inference

```
procedure multi.window.protocol.site.i (N, t, X⃗, X⃗-order);
/* N:        Estimated total number of distributed random variates;
   t:        number of random variates to be selected;
   L & U:    Lower & upper bounds of the domain of random variates;
   X⃗:        random variates generated by the local processor i
             (= (x_1, .... x_t));
   x_i-order: Order of X⃗ among the N distributed random variates;
   W⃗:        vector of search windows (= (w⃗_1, .... w⃗_t),
             w⃗_i = (w_i,1, w_i,2));
   S⃗ & D⃗:    Collision-detection vectors;
   Q⃗:        = (q_1..... q_t). q_i represents the estimated order of
             a random variate in window i;
   T:        Termination indicator */
begin
   for i=1 to t do parallel
      begin
      s_i ← 0; d_i ← 0; q_i ← 1; x_i-order ← 1;
      end;
   w_t+1,1 ← L; T ← U; done ← false;
   while ((not done) and (not all x_i-order > t)) do
      begin
      /* Determine the transmission windows */
      window (W⃗, Q⃗, D⃗, T);
      /* Transmit to k'th channel if w_k,1 < x_i ≤ w_k,2 */
      transmit(X⃗, W⃗);
      observe(S⃗ , D⃗): /* Detect outcome of contention. */
      /* Update current order of each window and X⃗-order */
      for k=1 to (t+1) do
         begin
         q_k ← q_k + Σ_{j=1}^{k-1} s_j;
         for i=1 to t do parallel
            if (x_i ≥ w_k,2) then x_i-order ← q_k + s_k;
         end;
      /* Update search range by resetting termination indicator */
      i ← 0; r ← 0;
      while ((r < t) and (i < t)) do
         begin
         i ← i+1; r ← q_i+2*Σ_{j=1}^{i} d_j;
         end;
      if (i < t) then
         begin
         T ← w_i,2;
         /* Termination ? */
         if ⌊Σ_{j=1}^{i} d_j = 0⌋ then done ← true;
         end;
      end;
end multi-window-protocol.
```

Figure 2. Multi-window protocol for ordered selections

minimum is less than w and the second minimum is greater than w after a contention is resolved. Based on this value, the maximum-likelihood estimate of the channel load can be estimated. Let y_1 be the minimum and y_2 be the second minimum. After the i'th contention, the window (a, w(i)] isolates the minimum successfully. The maximum-likelihood function for the estimated channel load $\hat{n}(i)$ can be formulated as

$$L(\hat{n}(i),w(i),a) = Pr(a<y_1<w<y_2) \qquad (6)$$
$$= \hat{n}(i)w(i)(1-w(i))^{\hat{n}(i)-1}$$

$L(\hat{n}(i),w(i),a)$ is maximized at

$$\hat{n}(i) = \left\lfloor \frac{-1}{\ln(1-w(i))} \right\rfloor \quad 0<w(i)<1 \qquad (7)$$

The estimated number of contending stations in the (i+1)'th contention can be obtained by adding to $\hat{n}(i)$ the difference between

```
procedure window (W⃗, Q⃗, D⃗, T);
/* N:        Total number of distributed random variates;
   t:        Number of random numbers to be selected;
   L & U:    Lower & upper bounds on the range of random variates
             supplied from the calling function;
   W⃗:        vector of search windows (= (w⃗_1, .... w⃗_t),
             w⃗_i = (w_i,1, w_i,2));
   new-W⃗:    Temporary storage for new search windows;
   D⃗:        Collision indicator;
   Q⃗:        Estimated order of random variates in the windows;
   T:        Termination marker */
begin
   i ← 1; j ← 1;
   while (i ≤ t and w_j,2 ≤ T) do
      begin
      while (d_j=0) do j ← j+1;
      if (j≤t) then
         begin
         /* Allocate two busses to resolve a collided interval */
         new-w_i,1 ← w_j,1;
         new-w_i,2 ← (w_j,1+w_j,2)/2;
         new-q_i ← q_j;
         i ← i+1;
         new-w_i,1 ← new-w_i-1,2;
         new-w_i,2 ← w_j,2;
         new-q_i ← q_j;
         i ← i+1; j ← j+1;
         end;
      else begin /* Search into unsearched interval */
         for k=i to t do
            begin
            new-w_i,1 ← w_i,2+(k-i)(U-w_i,2)/(N-q_t-2*Σ_{j=1}^{t} d_j);
            new-w_i,2 ← new-w_i,1+(U-w_i,2)/(N-q_t-2*Σ_{j=1}^{t} d_j);
            new-q_i ← q_t;
            end;
         i ← t+1;
         end;
      end;
   W⃗ ← new-W⃗; Q⃗ ← new-Q⃗;
end.
```

Figure 3. A heuristic window-control procedure

the possible expected arrivals after the i'th contention.

The above maximum-likelihood estimation does not use a priori information. To improve the accuracy, windows that successfully isolate the minimum in previous contentions can be incorporated into the estimation. A technique in time-series analysis called Auto-Regressive-Moving-Average (ARMA) model can be applied to obtain an estimated window based on all previous windows, w(1), w(2), ... w(i). As an example, $w_{mv}(i)$ can be computed recursively as

$$w_{mv}(i) = \left\lfloor w_{mv}(i-1)+\frac{w(i)}{2} \right\rfloor \qquad (8)$$

w(i) in Eq. (7) can be replaced by $w_{mv}(i)$ above in estimating the channel load. An explicit way of using a priori information and collided intervals in estimation is discussed elsewhere [JuL85].

4. CORRECTNESS AND PERFORMANCE EVALUATION

In this section, we will prove that the proposed multi-window protocol correctly identifies the t smallest variates. The performance of the protocol is also discussed.

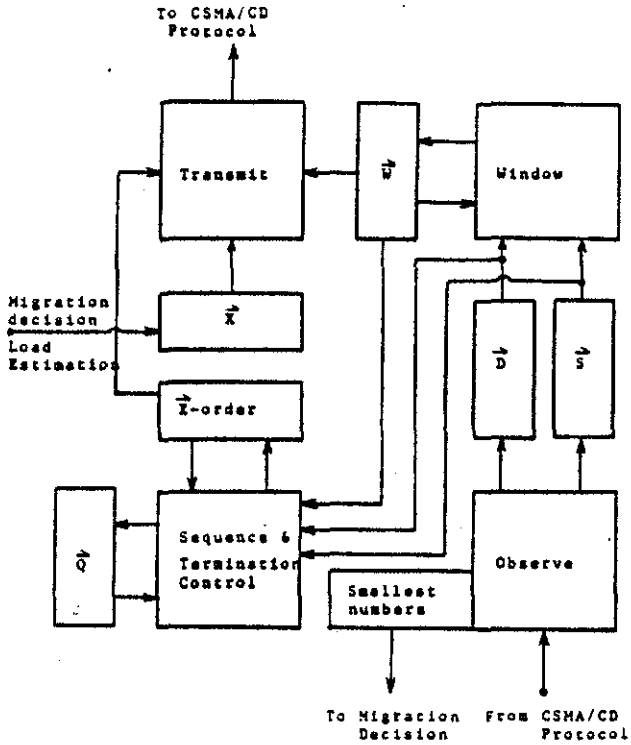

Figure 4a. Hardware architecture to support the multi-window protocol.

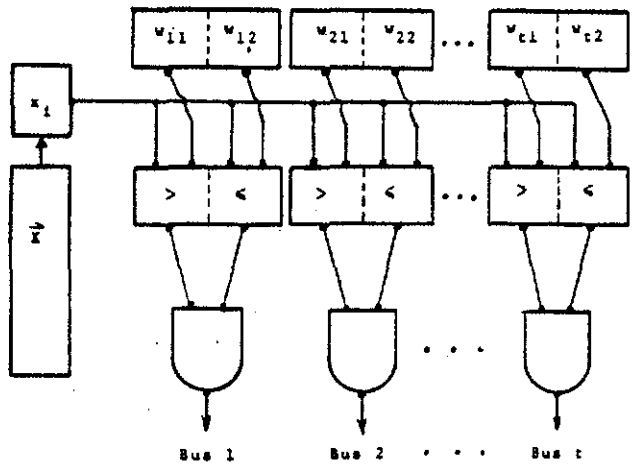

Figure 4b. Design of the *transmit* function in the multi-window protocol.

4.1. Correctness

First, we show that an ordered selection terminates in a finite number of steps.

Lemma 1: The multi-window protocol terminates in a finite number of steps if the random numbers to be selected are separable. Two random numbers, y_i and y_j, are separable if $\frac{1}{N(y_i-y_j)}$ is finite.

Proof: The procedure terminates when all disjoint intervals below the termination indicator have been resolved. It is necessary to show that there are a finite number of such intervals and that these intervals were obtained from partitioning the search range in a finite number of steps. To search an unresolved interval, the window control determines sub-intervals of non-zero

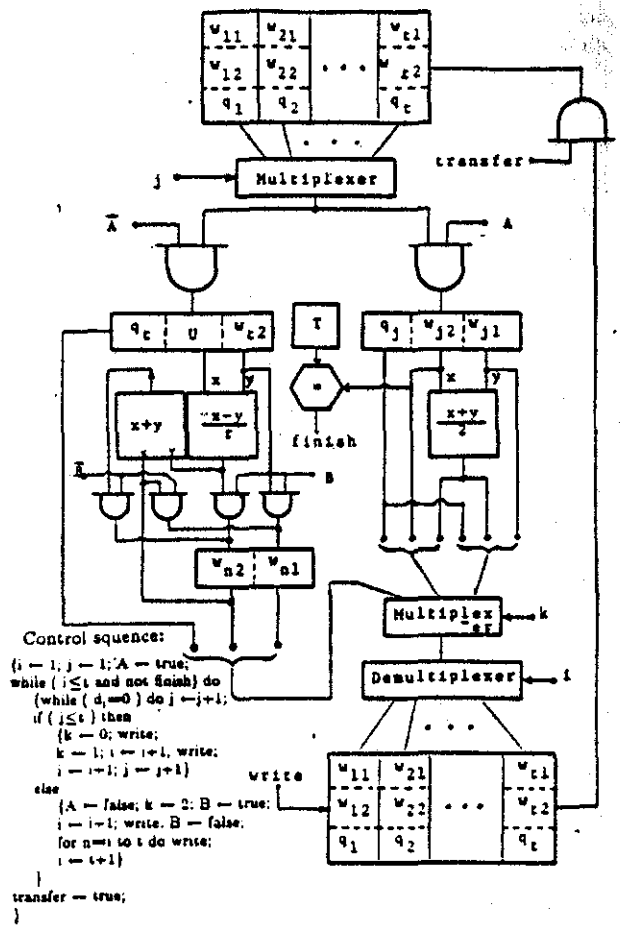

Figure 4c. Design of the *window* function in the multi-window protocol.

size as contention windows and partitions the search range into a finite number of sub-intervals. These sub-intervals may be resolved or remain unresolved after a contention. If a sub-interval remains unresolved, it is split into two halves of finite sizes. The maximum number of steps to separate any two random numbers is

$$k = \left\lfloor \log_2 \frac{1}{N \cdot \delta} \right\rfloor \qquad (9)$$

where $\delta = \min\{|y_i - y_j|, 1 \leq i, j \leq t+1, \text{and } i \neq j\}$. Since δ is finite, so is k. Thus the procedure terminates in a finite number of steps. □

The following lemma proves the correctness of the termination condition.

Lemma 2: All the intervals below the termination indicator are resolved when the multi-window protocol terminates.

Proof: After the first step of contention, there are two possible outcomes:

$$(q_t - 1) + 2 \cdot \sum_{j=1}^{t} d_j < t, \quad \text{or} \qquad (10a)$$

$$(q_t - 1) + 2 \cdot \sum_{j=1}^{t} d_j \geq t \qquad (10b)$$

Note that q_t is the approximated order of the variates in the t'th window before it is updated, and that the d_i's are the outcomes of the current contention. Since $q_t \geq 1$, we have $2 \cdot \sum_{j=1}^{t} d_j < t$ in the

first case, which implies that the number of unresolved intervals below the termination indicator is less than the number of busses available. Hence there are a sufficient number of busses to be allocated, and the remaining busses may be used to extend the resolution process into the unsearched range.

Since intervals are repeatedly partitioned as contention proceeds, the number of collided intervals may increase, and the second case may happen eventually. This is the case in which there are more intervals to be resolved than the number of available busses, and there are more than t variates below the termination indicator. There exists an index k, $1 \leq k \leq t$, such that

$$(q_k-1)+2\cdot\sum_{j=1}^{k} d_j = \begin{cases} t & \text{success in the k'th bus;} \\ t+1 & \text{collision in the k'th bus.} \end{cases} \qquad (11)$$

Accordingly, the termination indicator will be moved to the upper bound of the k'th window in this case. The total number of busses required to resolve all collided intervals within this range is less than t if either $q_k > 1$ or $2\cdot\sum_{j=1}^{k} d_j = t$. On the other hand, there will be an unallocated interval if $2\cdot\sum_{j=1}^{k} = t+1$. This interval will be excluded if the outcome of the next contention step shows that there are more than t numbers below the termination indicator, i.e., one or both of the following conditions are not satisfied.

$$2\cdot\sum_{j=1}^{t-1} d_i = t-1, \quad \text{and} \quad s_t+d_t=0 \qquad (12)$$

In contrast, if the above conditions are satisfied, then the t'th bus will be allocated to search this interval in the next contention step. Hence every sub-interval below the termination indicator will be searched until all sub-intervals are resolved. □

The correctness of proposed multi-window protocol can be summarized in the following theorem.

Theorem 2: The multi-window protocol with the proposed heuristic window control performs an ordered selection correctly.

Proof: In Lemma 1, we have shown that the protocol terminates in a finite number of steps. According to Lemma 2, all sub-intervals below the termination indicator are resolved. From the way the termination indicator is set, it is easy to show that there are at least t numbers being isolated in these sub-intervals. Since resolved sub-intervals are disjoint and follow a linear ordering relation, the numbers isolated in these sub-intervals can be ordered correctly. □

4.2. Performance

Simulations have been conducted to evaluate the performance of the multi-window protocol. The simulator was coded in F77 and run on a VAX 11/780 computer. In the simulator, each processor generates a random number uniformly distributed in [0,1]. A collision-detection mechanism is modeled by a counter that counts the number of random variates in a given sub-interval. Different combinations of N and t were evaluated, each of which was run a number of times with different seeds until a 95% confidence interval of less than 0.2 was obtained. The simulation results are shown in Tables 1 and 2. They show that the average number of contention steps to identify the t smallest variates out of N random variates can be approximated by $\alpha(N,t)$ ($=0.8\cdot\log_2 t + 0.2\cdot\log_2 N + 1.2$) with less than 5% error. This approximation was obtained under the assumption that t busses are employed when t numbers are to be identified.

N=20										
t	2	4	6	8	10	12	14	16	18	20
$\bar{C}$	2.97	3.65	4.20	4.49	4.78	4.99	5.20	5.33	5.51	5.53
$\alpha(N,t)$	2.86	3.66	4.13	4.46	4.72	4.93	5.11	5.26	5.40	5.52
N=100										
t	10	20	30	40	50	60	70	80	90	100
$\bar{C}$	5.19	5.77	6.46	6.65	7.00	7.15	7.37	7.56	7.67	7.75
$\alpha(N,t)$	5.19	5.96	6.46	6.79	7.04	7.26	7.43	7.59	7.72	7.84
N=500										
t	50	100	150	200	250	300	350	400	450	500
$\bar{C}$	7.39	8.12	8.42	8.85	8.96	9.14	9.51	9.87	9.89	10.0
$\alpha(N,t)$	7.51	8.31	8.77	9.11	9.36	9.57	9.75	9.91	10.0	10.2

Table 1. Average number of contention steps ($\bar{C}$) to identify t smallest numbers among N distributed random variates using the proposed ordered selection protocol (N is fixed; $\alpha(N,t) = 0.2\log_2 N + 0.8\log_2 t + 1.2$ is given here for comparisons).

t=10										
N	30	60	90	120	150	180	210	240	270	300
$\bar{C}$	4.88	4.93	5.09	5.13	5.18	5.20	5.40	5.50	5.54	5.58
$\alpha(N,t)$	4.84	5.04	5.16	5.24	5.31	5.36	5.40	5.44	5.48	5.51
t=50										
N	50	100	150	200	250	300	350	400	450	500
$\bar{C}$	6.77	7.04	7.22	7.26	7.28	7.40	7.37	7.47	7.60	7.60
$\alpha(N,t)$	6.85	7.05	7.17	7.25	7.31	7.37	7.41	7.45	7.48	7.51

Table 2. Average number of contention steps ($\bar{C}$) to identify t smallest numbers among N distributed random variates using the proposed ordered selection protocol (t is fixed; $\alpha(N,t) = 0.2\log_2 N + 0.8\log_2 t + 1.2$ is given here for comparisons).

5. DISTRIBUTED CHANNEL ALLOCATION WITHOUT PRIORITY

In allocating channels without priority, a contention bus can be seen as a server with homogeneous customers. To transmit its packet, a processor has to contend for bus access. The question is how many busses does it have to contend and which busses to use. The number of busses to contend may range from one to all. If the strategy is to allow a processor to contend only one bus at a time, then the issue of determining the bus to contend is the problem of load balancing. If the strategy is to contend for all busses, then there may be redundant winners and may result in a poor performance. It will be shown in Section 5.2 that this redundancy is not crucial.

5.1. Channel Allocation by Load Balancing

Probabilistic and state-dependent routing has been proposed for balancing jobs among multiple servers. In probabilistic routing, packets are routed to a channel according to the branching probabilities. For example, a processor may route its requests to Channel 1 with probability 0.2, to Channel 2 with probability 0.3, and to Channel 3 with probability 0.5. When a new request arrives, the processor may decide to route the request to Channel 2 based on the outcome of a generated random number. Since channels are chosen probabilistically, it is possible that a processor may select a channel that is already heavily loaded. Such an improper routing usually leads to an imbalance of loads among the channels and a poor channel utilization. To improve the performance of probabilistic routing, the branching probabilities may be optimized with respect to statistics collected in real time. These statistics include request generation rates, bus speeds, and transmission times [NiH85]. The objective is to balance the

channel loads such that the traffic intensities to different channels are equal. Probabilistic routing is usually unable to adapt to instantaneous changes in workload and performs poorly when the variance of workloads is large.

On the other hand, state-dependent routing can cope with the problem by determining the routing on a per-packet basis according to the channel load. A request is routed to the channel with the minimum load. In general, it is difficult to implement state-dependent routing in a multi-bus network since the channel loads are unknown. However, the bus load can be estimated easily by the interval-resolution scheme in the proposed protocol.

Simulations were conducted to evaluate state-dependent routing with load estimation (Figure 5). As a comparison, the analytical results of probabilistic routing are also plotted. These results show that as high as 60% of the channels are wasted in probabilistic routing, but is reduced to less than 5% in state-dependent routing. In contrast to probabilistic routing, the percentage of channels wasted in state-dependent routing decreases as the number of channels in the network increases.

Note that there may be more than one request generated before the channel-load information is updated. These requests are routed to the same channel if they originated from different processors. The simulation results also indicate that the effect of such an improper routing is minor.

5.2. Non-priority Channel Allocation by Redundant Contentions

To reduce the probability of a channel being wasted, a processor may contend for all available channels and selects one from those it wins. The fact that a processor may win more channels than it needs seems to be the major disadvantage of this approach. The following observation reveals that the effect of redundant allocations is small. Suppose that N processors are contending for t busses. A bus will be allocated to a processor with probability $\frac{1}{N}$. The expected number of busses allocated to a processor is

$$\sum_{n=0}^{t} n \cdot \binom{t}{n} \left(\frac{1}{N}\right)^{n} \left(1-\frac{1}{N}\right)^{t-n} = \frac{t}{N} \qquad (13)$$

The probability that each processor is allocated at least one channel is

$$1 - \left(1 - \frac{1}{N}\right)^{t} \approx 1 - 1 + \frac{t}{N} = \frac{t}{N} \qquad N \to \infty \qquad (14)$$

Using Eq. (14) and assuming that exactly one channel is allocated to each winning processor, the expected number of channels allocated to each is t/N, which is the same as that of Eq. (13). This shows that redundant allocation is not likely to happen (when N is large).

6. CONCLUDING REMARKS

A multi-window protocol for supporting priority-based channel allocation is studied in this paper. The protocol utilizes the collision-detection mechanism to resolve interval status on the contention busses. It can identify t packets of the highest priority among N contenders in about $0.8 \log_2 t + 0.2 \log_2 N + 1.2$ contention steps. A hardware architecture for implementing the proposed protocol is also presented.

The load-estimation capability of the proposed protocol allows a state-dependent routing strategy to be adapted in non-priority channel allocation. The fraction of wasted channels is reduced to less than 5% in state-dependent routing. Contending for multiple channels is also shown to be a promising non-priority channel-allocation strategy.

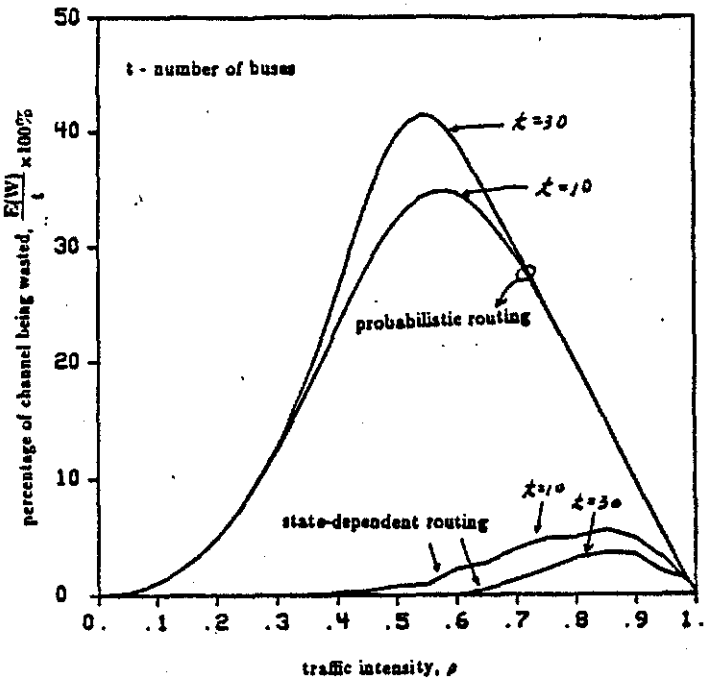

Figure 5. Comparison of probabilistic routing and state-dependent routing with load estimation.

REFERENCES

[ChK79] Y. C. Chow and W. Kohler, "Models for Dynamic Load Balancing in a Heterogeneous Multiple Processor System," *IEEE Trans. on Computers*, Vol. 28, No. 5, May 1979, pp. 334-361.

[GoF83] Y. I. Gold and W. R. Franta, "An Efficient Collision-Free Protocol for Prioritized Access-Control of Cable Radio Channels," *Computer Networks*, Vol. 7, North-Holland, 1983, pp.83-98.

[JuW84] J. Y. Juang and B. W. Wah, "Unified Window Protocol for Local Multiaccess Networks," *Proc. IEEE INFOCOM 84*, San Francisco, CA, April 1984, pp.97-104.

[Jua85] J. Y. Juang, *Resource Allocation in Computer Networks*, Ph.D. Thesis, Purdue University, August 1985.

[JuL85] J. Y. Juang and C. C. Lee, "Load Estimation in CSMA/CD Networks," submitted to the 6th International Conference on Distributed Computing Systems, Cambridge, MA, May 1986.

[MaR82] M. A. Marsan and D. Roffinella, "Nonpersistent M-CSMA protocols for Multichannel Local Area Networks," *Proc. 7'th Conf. on Local Computer Networks*, Minneapolis, MN, 1982.

[NiL83] L. M. Ni and X. Li, "Prioritizing Packet Transmission in Local Multiaccess Networks," *Proc. of Eighth Data Communications Symposium*, Cape Cod, MA, 1983.

[NiH85] L. M. Ni and K Hwang, "Optimal Load Balancing in a Multiple Processor System with Many Job Classes," *IEEE Trans. Software Engineering*, Vol. SE-11, No.5, May 1985, pp. 491-496.

[Sha83] N. Schacham, "A Protocol for Preferred Access in Packet-switching Radio Networks," *IEEE Trans. Communications*, Vol. COM-31, No.2, Feb. 1983, pp. 253-264.

[Tob82] F.A. Tobagi, "Carrier Sense Multiple Access Networks with Message-Based Priority Functions," *IEEE Trans.*

Communications, Vol. COM-30, No.1, Jan. 1982.

[Tow80] D. Towsley, "Queueing Network Models with State Dependent Routing," *JACM*, Vol. 27, No. 2, April 1980, pp. 323-337.

[WaJ83] B. W. Wah and J. Y. Juang "An Efficient Protocol for Load Balancing on CSMA/CD Networks," *Proc. 8'th Conf. on Local Computer Networks*, IEEE, Oct. 1983, pp. 55-61.

[WaJ85] B. W. Wah and J. Y. Juang, "Resource Scheduling for Local Computer Systems with a Multiaccess Network," *IEEE Trans. Computers*, Vol. C-34, No. 12, Dec. 1985, pp. 1144-1157.

APPENDIX A

In this appendix, rules in the heuristic window control are analyzed.

First, we show that partitioning a collided interval into two equal halves is a good heuristic rule. It has been proved that the binary-divide strategy is optimal for resolving a collided interval if there are two variates uniformly distributed in this interval [WaJ83]. Suppose that the size of the unsearched interval is u and that the estimated number of variates falling in this intervals is $\hat{r}$. Then the size of the window to be searched initially is $u/\hat{r}$. Let Z be the random variable representing the number of y_i's in such an interval. Z has a binomial distribution since the y_i's are uniformly distributed and will fall in the window with probability $1/\hat{r}$. The expected value of Z can be expressed as

$$E(Z) = \sum_{p=1}^{\hat{r}} p \binom{\hat{r}}{p} \left(\frac{1}{\hat{r}}\right)^p \left(1-\frac{1}{\hat{r}}\right)^{\hat{r}-p} \qquad (A1)$$

The values of E(Z) with respect to different $\hat{r}$'s are given in Table A1. They show that E(Z) is about 2.3 and less than 2.38 up to a very large value of $\hat{r}$.

A collided window contains less than 2.38 variates on the average since the expected number of y_i's in a partitioned window is always less. The arguments above indicate that a collided window often contains two variates, and that the binary-divide rule is a good approximation to the optimal window control.

The following lemma and theorem show that the heuristic rule of determining the interval size in the unsearched range is also an efficient strategy.

Lemma A1: Let p be a natural number, and $y_i > 0$. If $\sum_{i=1}^{p} y_i = c$, then $\prod_{i=1}^{p} y_i$ is maximized when the y_i's are equal. That is, $y_i = c/p$, i=1, ... p.

Proof: The lemma is proved by mathematical induction. The induction basis (p=1) is trivial. Consider the case in which p is greater than one. Assume that

$$\sum_{i=1}^{p-1} y_i = \beta c \quad \text{where } 0 < \beta < 1 \qquad (A2)$$

Accordingly, $y_p = (1-\beta)c$. If p is equal to 2, then

$$y_1 * y_2 = c^2\beta(1-\beta) \qquad (A3)$$

The RHS of Eq. (A3) is maximized when $\beta = 0.5$, so the lemma is true for p = 2.

Assume that the lemma is true for p=m, and consider p=m+1. Let $y_{m+1}=(1-\beta)c$ and $\sum_{i=1}^{m} y_i=\beta c$. Then $\prod_{i=1}^{m} y_i$ is maximized at $y_i=\beta c/m$. Hence $\prod_{i=1}^{m+1} y_i$ can be rewritten as

$$\prod_{i=1}^{m+1} y_i = \left(\prod_{i=1}^{m} y_i\right) * y_{m+1} = \left(\frac{\beta c}{m}\right)^m (1-\beta)\, c \qquad (A4)$$

$\hat{r}$	E(Z)
2	2.0
5	2.312
10	2.326
20	2.329
100	2.337
500	2.346
1000	2.379

Table A1. E(Z), the expected number of random variates falling in a window of size $u/\hat{r}$, where $\hat{r}$ is the estimated number of random variates uniformly distributed in an unsearched interval of size u.

The RHS of Eq. (A4) is maximized at $\beta=\frac{m}{m+1}$, which yields $y_i=\frac{c}{m+1}$ for i=1, ... m+1. Therefore, the lemma is also true for p=m+1. □

Theorem A1: Let the total number of y_i's distributed in the unsearched range [0,u) be n, and the boundaries of the i'th window be $[w_{i-1}, w_i)$, i = 1, ... t. Then the probability that all t of the y_i's are isolated in one contention step is maximized if $w_i - w_{i-1} = u/n$, i = 1, ... t.

Proof: In general, the joint probability density function of the ordered statistics, $y_1, \ldots, y_{t+1}$, is

$$f_{y_1 \ldots y_{t+1}}(x_1, \ldots, x_{t+1}) \qquad (A5)$$
$$= \frac{n!}{(n-t-1)!} g(x_1) \ldots g(x_{t+1}) (1-G(x_{t+1}))^{n-t-1}$$

where g(x) and G(x) are the probability density function and distribution function of the parent distribution, respectively. For a uniform distribution, we have g(x)=1 and G(x)=x. So,

$$f_{y_1 \ldots y_{t+1}}(x_1, \ldots, x_{t+1}) = \frac{n!}{(n-t-1)!}(1-x_{t+1})^{n-t-1} \qquad (A6)$$

A selection is successful when $y_i \in (w_{i-1}, w_i)$, i = 1, ... t, and $y_{t+1} \in [w_t, 1)$, implying that $y_{t+2}, \ldots, y_n \in [w_t, 1)$. The success probability may be expressed as

$$\Pr\left(y_i \in [w_{i-1}, w_i),\ i=1, \ldots, t, \text{ and } y_{t+1} \in [w_t, 1)\right) \qquad (A7)$$
$$= \int_0^{w_1} \cdots \int_{w_{t-1}}^{w_t} \frac{n!}{(n-t-1)!}(1-x_{t+1})^{n-t-1}\, dx_{t+1} \ldots dx_1$$
$$= \frac{n!}{(n-t)!}(1-w_t)^{n-t} \prod_{i=1}^{t} (w_i - w_{i-1})$$

Note that the value of $\prod_{i=1}^{t}(w_i-w_{i-1})$ depends on w_t and the partitioning of the interval $[0,w_t)$. It follows from Lemma A1 that $\prod_{i=1}^{t}(w_i-w_{i-1})$ is maximized when $w_i - w_{i-1} = w_t/t$. Substituting it into Eq. (A7) yields

$$\Pr\left(y_i \in [w_{i-1}, w_i), i=1, \ldots, t, \text{ and } y_{t+1} \in [w_t, 1)\right) \qquad (A8)$$
$$= \frac{n!}{(n-t)!}(1-w_t)^{n-t} \left(\frac{w_t}{t}\right)^t$$

The RHS of Eq. (A8) is maximized at $w_t = t/n$. This leads to the results of $w_i = i/n$. Lastly, the w_i's have to be adjusted by a factor of u if the random variates are distributed in the interval [0,u). Therefore, $w_i - w_{i-1} = u/n$. □