Mathematical and Computational Applications, Vol. 15, No. 2, pp. 156-165, 2010.
© Association for Scientific Research

A NEW LOOK-AHEAD ALGORITHM TO IMPROVE THE PERFORMANCE OF OMEGA NETWORKS

Abdulaziz S. Almazyad
Department of Computer Engineering
College of Computer and Information Sciences
P.O Box 51178, Riyadh 11543
King Saud University
mazyad@ksu.edu.sa

Abstract- In this paper, we concentrate on the internal blocking problem related to Omega Network which is a type of Multistage Interconnection Networks (MINs). We introduce a new algorithm to alleviate the internal blocking problem, the function of this look-ahead algorithm is to decide which packet to block in case of internal blocking. The performance results indicate that the introduced algorithm decreases the internal blocking problem significantly. Since the computations involved in the introduced algorithm are so little, it can be implemented in real time effectively.

Key words- MINs, Omega networks, Internal blocking, Look-ahead algorithm.

1. INTRODUCTION

Omega Network is a type of MINs, in which, $a = b = 2$. The general architecture of an Omega Network consists of a few stages of a number of switching elements (SEs), each stage is connected with the next stage with a specific interconnection called: *perfect shuffle*, except the last stage, in which the identity permutation is used. The perfect shuffle permutation is performed by rotating the binary representation of the input link one bit to the left[1,2,3]. So, if the binary representation of an input link is :

$$x_{n-1}x_{n-2}x_{n-3}\ldots\ldots x_1x_0$$

Then the packet will be routed to a destination with a binary format :

$$x_{n-2}x_{n-3}\ldots\ldots x_1x_0x_{n-1}$$

The perfect shuffle connection of Omega Networks simply divides the *N*-channels into two halves, which are then interleaved perfectly. Basically, the 2×2 SE has two configuration, straight or exchange as illustrated in Fig. 1.1



(a) Straight (b) Exchange

Figure 1.1 Configurations of a switching element

In general, Omega Network with *N* inputs consists of $n = \log_2 N$ identical stages, and each stage consists of a perfect shuffle connection followed by *N/2 2 x2* switch elements. Since each switch element has two configurations, namely, straight and exchange; therefore number of realizable permutations by the network is [1]:

$$\text{Realizable Permutations} = 2^{\frac{N.\log N}{2}}$$

For example, an 8×8 omega network has 4096 realizable permutations. Since total permutations is *N!* , only 10% of an 8×8 omega network permutations are realizable, and 90% of permutations cause internal blocking. For a 16×16 omega network, 99.98% of the permutations cause internal blocking! and it get worse with random input omega networks. So, we need algorithms to alleviate the problem of internal blocking. Fig 1.2 shows an omega network with $N = 8$.

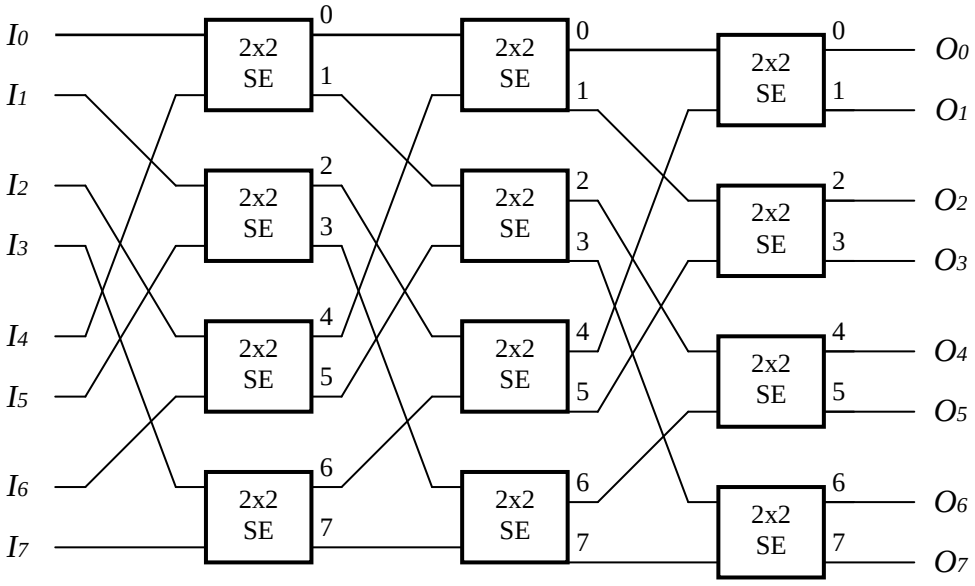


Figure 1.2 An 8x8 Omega Network, Consisting of 3-stages, with each stage consisting of a perfect shuffle followed by 4 switches

MINs are used because they are less expensive, easy to control, have low delay and support large scale of inputs/outputs. One of applications of MINs is in processor to memory communication in parallel multiprocessing systems, in which, they allow a direct link between any processor to any memory module so the processor can access any memory module with a very small number of communications or accessing conflict.

Internal blocking can be eliminated by increasing the number of switch elements [4], the number of stages [4], or the size of the switch element [5]. However, all these techniques increase the cost and delay of such networks. Therefore, in this paper we attempt to use as few switch elements as possible, while maintaining the full accessibility by using our new technique.

2. RESOLVING INTERNAL BLOCKING USING A LOOK-AHEAD ALGORITHM

Internal blocking occurs in an omega network switch when two input packets attempt to go to the same output link. When an internal blocking occurs, one of the two conflicting packets must be blocked, after that, the other packet can be passed without conflict. Normaly, when internal blocking occurred, it has been resolved by randomly blocking one input packet and passing the other input packet, we call this *Random blocking*. One may always blocks the lower input and passes the other one, or always blocks the upper input and passes the other one, we call this *fixed blocking.*
Since each of the two contending input packets has different routing path, we expect that the choice of which packet that should be blocked will affect the internal blocking in the next stages and thus will affect the network performance. In this paper, an algorithm has been introduced to increase the performance of the omega networks. The function of this algorithm is to decide which packet should be blocked when an internal blocking occurs.

2.1 Effect of the decision of which packet to block

The selection of the packet that should be blocked in the case of internal blocking affects the internal blocking count in the next stages remarkably. The blocking selection effect may be illustrated by an example.

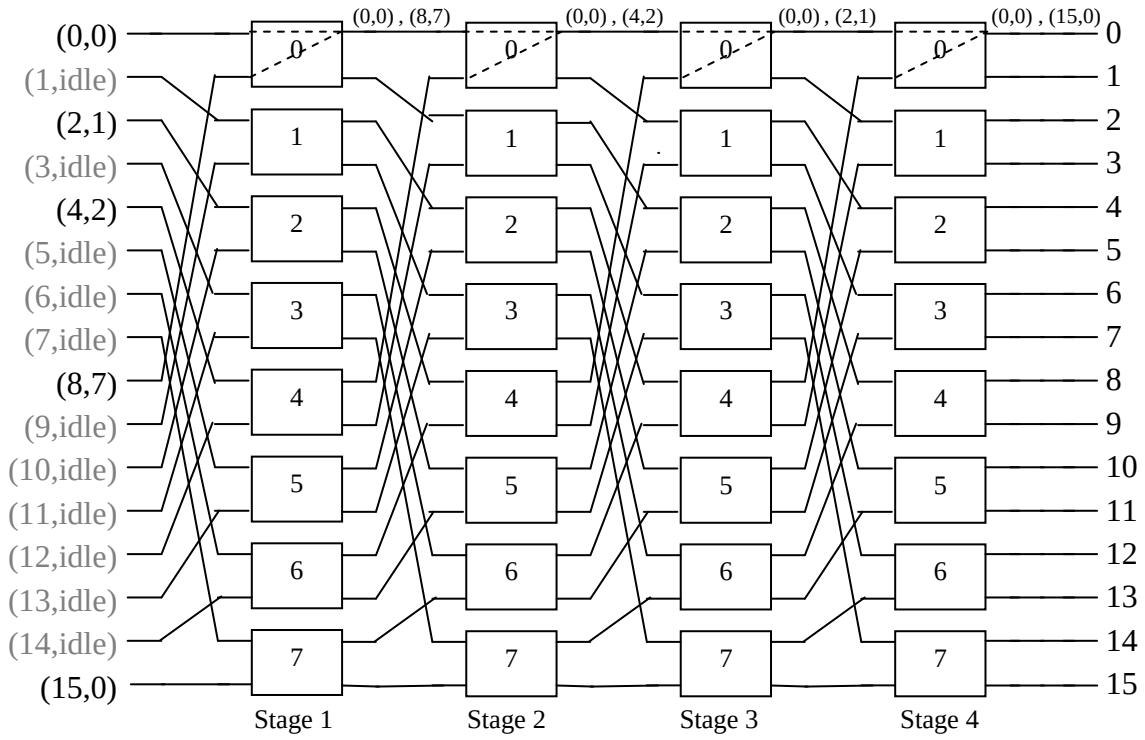


Figure 2.1 Example illustrates the effect of the choice of the packet that should be blocked in case of internal blocking

Fig 2.1 shows an example of a 16×16 omega network. In this example, inputs are idle except the inputs 0, 2, 4, 8 and 15 which have the following destinations: 0,1,7, and 0 respectively. In Fig 2.1, we associate an ordered pair for each input link. The first number in the ordered pair is the input link index, whereas the second number indicates the destination.

In this example, there are four internal blocking cases, one in each stage. In stage 1, internal blocking occurs in switch 0 between inputs 0 and 8. In stage 2, internal blocking occurs in switch 0 between inputs 0 and 4. In stage 3, internal blocking occurs in switch 0 between the inputs 0 and 2. In stage 4, internal blocking occurs in switch 0 between inputs 0 and 15. Note that inputs 8, 4, 2 will go to the lower output port of the SEs in stages 2, 3, 4 respectivley. Hence if we block input 0 in stage 1, there will be no blocking between inputs 8 and 4 in stage 2, inputs 4 and 2 in stage 3, and inputs 2 and 15 in stage 4.

In this particular example, there are 5 possibilities based on the blocking choice in each stage, the possibilities are:

Table 2.1 Example shows the effect of blocking choice in the performance.

Case No	*Blocked Inputs at each Stage*				Total Blocking
	Stage 1	Stage 2	Stage 3	Stage 4	
1	8	4	2	15	4
2	8	4	2	0	4
3	8	4	0	None	3
4	8	0	None	None	2
5	0	None	None	None	1

Based on Table 2.1, it is clear that the decision of which input to block affects the total number of blocked inputs and thus affects the performance. In this particular example, the best performance can be achieved in case#5 when input 0 is blocked in stage 1, whereas the worst performance occurs in case#1 and case#2.

2.2 The Algorithm

Input packets are routed to their destinations in one *cycle*. The cycle involves routing the input packets through stage 1, stage2, .. up to the last stage. Before the cycle starts, the algorithm processes the input packets and block the input packets that cause more internal blocking. This pre-processing is done by first generating a weight and a contending set for each input packet. The generated weight assigned for a packet corresponds to the total number of input packets that causes internal blocking (contention) with that packet and the contending set indicates the indices of the contending packets. After all weights and sets are generated, a virtual cycle (inside the central controller) begins. During the cycle, when an internal blocking occurs, the algorithm selects the packet with highest weight for blocking and after that, it updates the input packets weights and sets related to the blocked packet.

To illustrate the algorithm, consider the following example: a 16×16 omega network is shown in Fig 2.2. In this example, the first number in the ordered pair is the

input link index, whereas the second number indicates the destination. In this example, there are 7 internal blocking cases, three in stage 1, two in stage 3 and two in stage 4. The algorithm will generate Table 2.2.

Table 2.2 The values computed by the algorithm for the example shown in Fig 2.2, before the cycle starts

Input Number	Weight	Contending Set
0	2	{8,15}
1	3	{6,7,9}
2	0	$\{\phi\}$
5	1	{15}
6	2	{1,9}
7	3	{1,9,15}
8	2	{0,15}
9	3	{1,6,7}
15	4	{0,8,5,7}

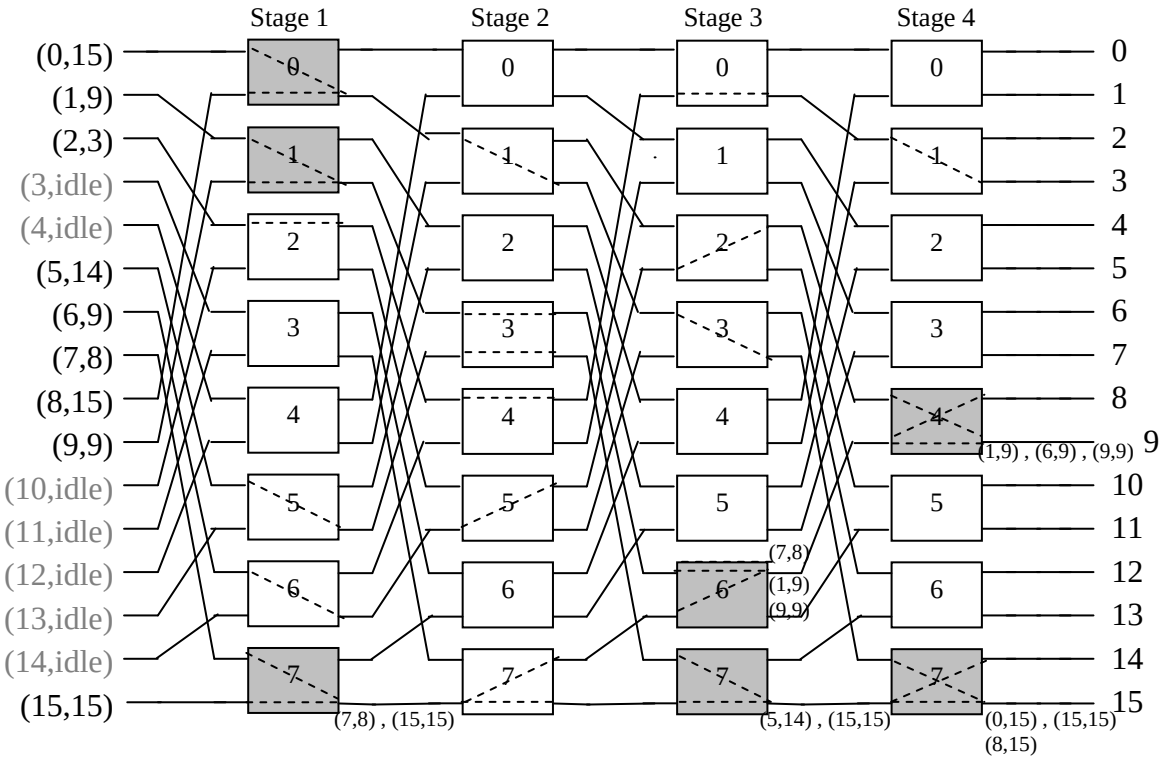


Figure 2.2 Example illustrates the algorithm

As in Figure 2.2, input 0 causes internal blocking with input 8 in switch 0 at stage 1 and with input 15 in switch 7 at stage 4. Input 1 causes internal blocking with input 9 in switch 1 at stage 1, with input 7 in switch 6 at stage 3 and with input 6 in switch 4 at stage 4. Input 2 does not cause any internal blocking.

After the algorithm generates Table 2.2, the cycle starts as follows:

In stage 1: there is an internal blocking caused by inputs 7 and 15 in switch 7. Since input 15 weight is 4 whereas input 7 weight is 3, input 15 is selected for blocking since it generates more contentions in the next stages, after blocking input 15, the algorithm updates all input weights related to input 15. In switch 0, inputs 0 and 8 have the same weight, we choose lower input (8) for blocking, in switch 1, inputs 1 and 9 have the same weight , we choose lower input (9) for blocking and we update the table of weights. Table 2.3 shows the weights of inputs after update.
In stage 2 : there is no blocking.

In stage 3 : the internal blocking in switch 6 between input 1 and input 7 is resolved by blocking input 1 since it's weight is 2 while the weight of input 7 is 1, then we update the weights. The weights become as shown in Table 2.4. Since all weights are zeros, no blocking will occur in the remaining stages (which is stage 4).

Table 2.3 The values computed by the algorithm for the example shown in Fig 2.2, after stage 1

Input Number	Weight	Contending Set
0	1	$\{\phi\}$
1	2	{6,7}
2	0	$\{\phi\}$
5	0	$\{\phi\}$
6	1	{1}
7	1	{1}
8	0	$\{\phi\}$
9	0	$\{\phi\}$
15	0	$\{\phi\}$

Table 2.4 The values computed by the algorithm for the example shown in Fig 2.2, after stage 3

Input Number	Weight	Contending Set
0	1	$\{\phi\}$
1	2	$\{\phi\}$
2	0	$\{\phi\}$
5	0	$\{\phi\}$
6	1	$\{\phi\}$
7	1	$\{\phi\}$
8	0	$\{\phi\}$
9	0	$\{\phi\}$
15	0	$\{\phi\}$

Generation of weights and contending sets:

The weight associated to an input packet is computed by counting all input packets that may cause internal blocking with the packet. The internal blocking between two packets can be detected by comparing partial bits of the destination bits of the two packets. The partial bits of the destinations the two packets that are used in the comparison depends on the stage in which the two packets may contend.

The procedure used to compute the weight and contending set of a packet from an input link *X* is as follows:

Assumptions:

- $N \times N$ omega network , number of stages = $n = \log_2(N)$.
- The weight of a packet from source link *X* is W_X .
- The contending set for a packet from source link *X* is *CS*.
- Binary representation for source link *X* is $x_{n-1}x_{n-2}\ldots x_2x_1x_0$.
- Packet *X* destination is *D* , or $d_{n-1}d_{n-2}\ldots d_2d_1d_0$ in binary.

Based on the binary representation $x_{n-1}x_{n-2}\ldots x_2x_1x_0$ of input *X*, the other *N-1* input packets can be partitioned into *n* sets, as follows :

$R_{X,1} = \{ \overline{x_{n-1}}x_{n-2}....x_2x_1x_0 \}$

$R_{X,2} = \{ 0\overline{x_{n-2}}\ldots x_1x_0 \ , \ 1\overline{x_{n-2}}\ldots x_1x_0 \}$

$R_{X,3} = \{ 00\overline{x_{n-3}}\ldots x_1x_0 \ , \ 01\overline{x_{n-3}}\ldots x_1x_0 \ , \ 10\overline{x_{n-3}}\ldots x_1x_0 , 11\overline{x_{n-3}}\ldots x_1x_0 \ \}$

……

$R_{X,n} = \{ 0..00\overline{x_0} \ , \ 0..01\overline{x_0} \ , \ 0..10\overline{x_0} \ , \quad \ldots\ldots \quad 1..11\overline{x_0} \ \}$

for all elements *Y* with destination *Z* in set $R_{X,i}$

if $(d_{n-1} = z_{n-1})$ and $(d_{n-2} = z_{n-2})$ ……. and $(d_{n-i} = z_{n-i})$ then

$W_X = W_X + 1;$ ** Contention occurs, so increment packet weight **

$CS = CS + \{Y\}$ ** Add *Y* to the contending list **

To illustrate how the procedure works, consider a 16×16 omega network. To evaluate the weight of an input packet *X*, the other inputs are partitioned in 4 sets, according to the stage in which the set elements may contend with the input *X* at. In stage 1, the input *X* is routed to the link $x_2x_1x_0d_3$ (according to omega network self routing feature), there is only one input that may contend with *X* at stage 1, this packet must come from link $\overline{x_3}x_2x_1x_0$. In stage 2, the input *X* is routed to the link $x_1x_0d_3d_2$, there are two packets which may contend with packet *X* at stage 2, they come from input links $0\overline{x_2}x_1x_0$ and $1\overline{x_2}x_1x_0$. Therefore, the sets associated with input X are:

$R_{X,1} = \{ \overline{x_3}x_2x_1x_0 \}$

$R_{X,2} = \{ 0\overline{x_2}x_1x_0 \ , \ 1\overline{x_2}x_1x_0 \}$

$R_{X,3}$ = { $00\overline{x_1}x_0$, $01\overline{x_1}x_0$, $10\overline{x_1}x_0$, $11\overline{x_1}x_0$ }

$R_{X,4}$ = { $000\overline{x_0}$, $001\overline{x_0}$, $010\overline{x_0}$, $011\overline{x_0}$, $100\overline{x_0}$, $101\overline{x_0}$, $110\overline{x_0}$, $111\overline{x_0}$ }

For example, the four sets associated with input 0000 are:

$R_{0,1}$ = { 1000 }

$R_{0,2}$ = { 0100 , 1100 }

$R_{0,3}$ = { 0010 , 0110 , 1010 , 1110 }

$R_{0,4}$ = { 0001 , 0011 , 0101 , 0111 , 1001 , 1011 , 1101 , 1111 }

Considering $R_{0,1}$, contention between 0000 and 1000 occurs only if there destination bits satisfy: $(d_3 = z_3)$. Considering $R_{0,2}$, contention between 0000 and 0100 occurs only if there destination bits satisfy: $(d_3 = z_3)$ and $(d_2 = z_2)$.Considering $R_{0,3}$, contention between 0000 and 0010 occurs only if there destination bits satisfy: $(d_3 = z_3)$ and $(d_2 = z_2)$ and $(d_1 = z_1)$. Considering $R_{0,4}$, contention between 0000 and 0001 occurs only if there destination bits satisfy: $(d_3 = z_3)$ and $(d_2 = z_2)$ and $(d_1 = z_1)$ and $(d_0 = z_0)$.

3. PERFORMANCE RESULTS

A simulation program has been implemented to measure the performance of the introduced algorithm. The simulation program has been executed for 10 arrival rates, 0.1, 0.2, 0.3, ..., 0.9 , 1.0. For each arrival rate, the algorithm has been executed for 2,500 random input connections. Furthermore, the previous process has been repeated 10 times to compute the confidence interval.

In the figures, there are four curves, each one corresponds to a blocking selection strategy, and the four selection strategies are:

- Random Select: when there is an internal blocking, select one packet randomly for dropping.
- Fixed Select: when there is an internal blocking, always select the upper (or always select the lower) packet for dropping.
- Best Select: when there is an internal blocking, use the algorithm to drop the packet that will cause *more* blocking in next stages. (This expected to result in a minimum dropping probability).
- Worst Select: when there is an internal blocking, use the algorithm to drop the packet that will cause *less* blocking in next stages. (This expected to result in a maximum dropping probability).

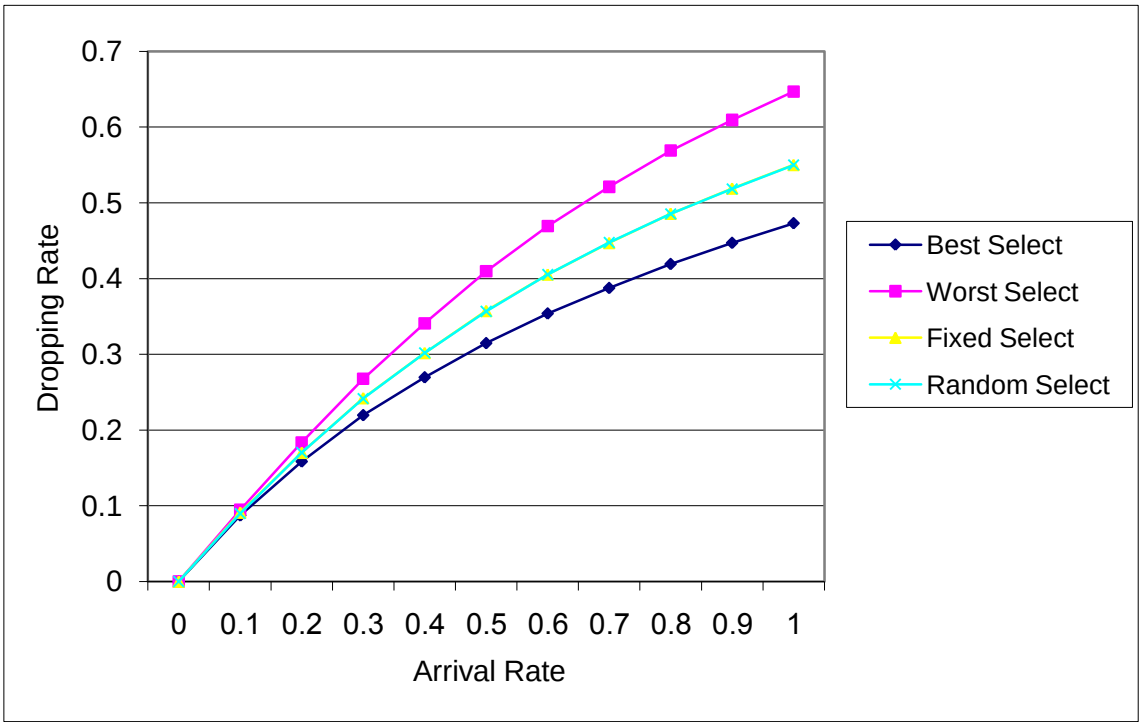


Figure 3.1 Dropping Probability. Comparing Look-ahead with Fixded/Random Droppping

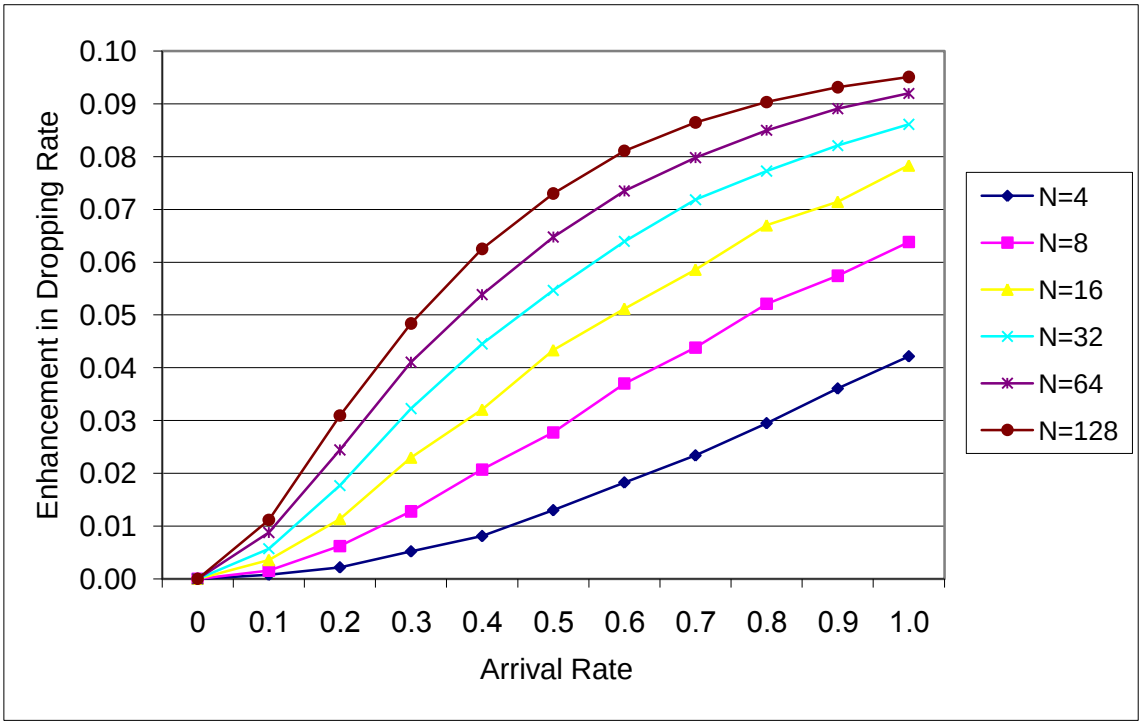


Figure 3.2 Effect of Look-Ahead Algorithm in decreasing Dropping Probability in an N x N Omega Network

Fig 3.1 shows that the gap between best and worst dropping probabilities increases with arrival rate. This shows that the algorithm becomes more effective as network load becomes heavy. As expected, random and fixed dropping probabilities are in the middle between the best and worst dropping probabilities, and they almost have the same values.

Fig 3.2 shows the dropping probability vs. network size. It is obvious that as the network size increases, dropping probability increases. Fig 3.2 indicates that the improvement in dropping probability as a result of using the algorithm becomes more effective as the network size increases. The reason is that we will have more stages and thus the look-ahead algorithm (which looks to next stages) becomes more effective.

4. CONCLUSION

To apply the algorithm we use the concept of centralized controller, in which, there is a central controller connected to Omega network switches. The central controler detects any internal blocking in a path between any source and any destination. If internal blocking is detected, the decision of which packet to block and which one to route affects the blocking probability and thus affects the network performance. Results obtained from simulation indicated that the look-ahead packet blocking algorithm introduced in this paper decreases the blocking probability. Hence we need fewer buffers to store the blocked packets.

5. REFERENCES

1. Mohammed Amer Arafah, Centralized Buffering and wavelength Conversion In Multistage Interconnection Networks, *A PhD Dissertation, University of Southern California, December 1997.*
2. Kuo-Chun Lee, Wavelength Conversion and Wavelength Routing in All-Optical Networks, *A PhD Dissertation , Electrical Engineering, University of Southern California, August 1994.*
3. Janak H. Patel, Performance Of Processor-Memory Interconnection for Multi-processors, *IEEE trans. On Computers, vol. **C-30**, Oct. 1981*.
4. G. Hugh Song, Asymmetric Dilation of Multwavelenegth Cross-Connect Switches for Low-Crosstalk WDM Optical Networks, *IEEE Journal of Lightwave Technology, vol.* **15**, *No.* **3**, *March* 1997.
5. Chiung-Shien et al., Extended Baseline Architecture for Nonblocking Photonic Switching, IEEE *Journal of Lightwave Technology, vol.* **15**, *No.***5**, *May* 1997.