Journal of King Saud University – Computer and Information Sciences (2011) **23**, 15–28

King Saud University

Journal of King Saud University – Computer and Information Sciences

www.ksu.edu.sa
www.sciencedirect.com

ORIGINAL ARTICLE

Optical omega networks with centralized buffering and wavelength conversion

Abdulaziz S. Almazyad

College of Computer and Information Sciences, King Saud University, Saudi Arabia

Received 14 March 2009; accepted 22 March 2010
Available online 8 December 2010

KEYWORDS

Optical omega networks;
Wavelength conversion;
Internal blocking;
Network performance;
MINs;
WDM

Abstract In this paper, we study the internal blocking problem in optical Multistage Interconnection Networks (MINs). We introduce algorithms to resolve internal blocking in optical MINs based on buffering and/or wavelength conversion. Since the computations involved in some of the introduced algorithms are so little, they can be implemented in real time effectively. A simulation program has been developed to verify the performance enhancement gained using the algorithms. Simulation results indicate that the developed algorithms effectively decreased the internal blocking and thus increased the network performance.

1. Introduction

Recently, significant developments have been made in photonic switching based on Wavelength Division Multiplexing (WDM). Therefore, researchers have used this technology to implement switches such as Crossbar or Multistage Interconnection Networks (MINs) in order to upgrade the performance of the available systems that use these networks (Liboiron-Ladouceur et al., 2008; Liboiron-Ladouceur and Bergman, 2007; Shares et al., 2007; Katangur et al., 2007; Al-Shabi and Othman, 2008; Lu and Zheng, 2007; Zhang and Yang, 2006, 2007; Ngo et al., 2007; Liu, 2008). Although the crossbar does provide a powerful communication capability, it is not economically feasible as the number of inputs/outputs becomes large. Moreover, due to the random nature of input arrivals, network utilization is much less than its capacity (Lee, 1994).

MINs are used because they are less expensive, easy to control, have low delay and support large scale of inputs/outputs. One of the applications of MINs is in processor to memory communication in a 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 communication or accessing conflicts (Patel, 1981).

These multistage interconnection networks are simple, inexpensive, and easy to build in a modular fashion. However, they have a problem of internal blocking. To alleviate the problem of internal blocking in MINs based on WDM, we can use buffers (Ngo et al., 2007) or wavelength converters (Zhang and

E-mail address: mazyad@ksu.edu.sa

Peer review under responsibility of King Saud University.
doi:10.1016/j.jksuci.2010.03.002

Production and hosting by Elsevier

Notations

N	number of omega network inputs. ($N = 2, 4, 8, 16,$ or $32 \ldots$)
n	number of stages in the omega network (equals to $\log_2 N$)
ω	number of maximum wavelengths available to a specific link
BufSize	number of available buffers
WCSize	total number of available wavelength converters in the network
Inputs	input stage data structure
Outputs	output stage data structure
I	index to represent input number in a given stage ($0 \leqslant I \leqslant N-1$)
J	index to represent wavelength number in a given input link ($0 \leqslant J \leqslant K-1$)
Switch	indicates the switch number for a given stage ($0 \leqslant Switch \leqslant \frac{N}{2}-1$)
Pkt(0)	packet coming from the upper input of a switch
Pkt(1)	packet coming from the lower input of a switch
P0	status of *Pkt(0)* destination. Values meanings are: $P0 = 0(1)$ means *Pkt(0)* destined to output upper (lower) link, $P0 = -1$ means *Pkt(0)* is idle
P1	status of *Pkt(1)* destination. Values meanings are: $P1 = 0$ (1) means *Pkt(1)* destined to output upper (lower) link, $P1 = -1$ means *Pkt(1)* is idle
Buffer.Size	number of currently buffered packets
TotBuff	total number of input packets that have been buffered
TotDrop	total number of input packets that have been dropped
TotConv	total number of wavelength conversion operations
TotInputs	total number of input packets that entered the omega network
TotOutputs	total number of outputs that have been processed
FreeWave	set of wavelengths that are not used by both inputs in a specific switch
FreeWaves(0, 0)	unused wavelengths in the upper input link that can only go to the output upper link
FreeWaves(0, 1)	unused wavelengths in the upper input link that can only go to the output lower link
FreeWaves(1, 0)	unused wavelengths in the lower input link that can only go to the output upper link
FreeWaves(1, 1)	unused wavelengths in the lower input link that can only go to the output lower link
CurWC_No	number of free wavelength converters that can be used $0 \leqslant CurWC\_No \leqslant WCSize$

Yang, 2006). Buffering and wavelength conversion techniques have been studied in detail in all-optical networks based on circuit switching and crossbar switches (Katangur et al., 2007) and (Al-shabi and Othman, 2008). In this paper, we focus on packet switching MINs.

Solving the problem of internal blocking in MIN's has been considered in many previous researches. Amer Arafah (1997) had developed new algorithms to resolve the internal blocking in MIN in Optical domain. The developed algorithms assume that all packets of any input channel should be forwarded to a single destination. This assumption is a non realistic and restricts the application of these algorithms.

In this paper, we relax this restriction and develop new algorithms such that packets of an input channel can have different destinations. We will introduce new algorithms to alleviate the problem of internal blocking based on centralized Wavelength Division Multiplexing (WDM) and/or centralized buffering. Using centralized WDM technology, the performance of omega networks can be improved.

The performance of optical omega network in the presence of buffering and/or wavelength conversion will be studied. Simulation will be implemented to study the effect of centralized buffering and wavelength conversion in decreasing the internal dropping (and thus improving the performance) in optical omega networks.

1.1. Prior research

The multistage interconnection networks have been studied intensively in the electrical domain. But due to the speed limitation of the electronic switching, many researches have been done over these networks in the optical domain. These researches investigate the implementation of MINs based on WDM by using either Guided-Wave Fabrics, which guide the propagation of the waves along a physically constructed path, or Free-Space Fabrics which utilize the spatial bandwidth without any predefined path by using mirrors, masks, polarized beam splitter (PBS), prism gratings, lenses, etc. the main objective of these researches is to implement such networks with very high bandwidth.

In this research, our goal is to implement a control unit for multistage interconnection network which can alleviate the problem of internal blocking by using buffering and wavelength conversion, and to improve the network performance.

We redefine internal blocking as two or more packets with the same wavelength trying to access a channel simultaneously. This problem can be eliminated by increasing the number of switch elements, the number of stages, or the size of the switch element. However, all these techniques increase the cost and delay of such networks. Therefore, in this research, we attempt to use as few switch elements as possible, while maintaining the full accessibility.

To alleviate the problem of internal blocking in MINs based on WDM, we can use buffers (Ngo et al., 2007) or wavelength converters (Zhang and Yang, 2006). The advantage of wavelength conversion over buffering is the ability to utilize the available channel bandwidth and to send a packet to its destination without waiting for the next switching cycle.

Buffering and wavelength conversion techniques have been studied in detail in all-optical networks based on circuit switching and crossbar switches (Katangur et al., 2007) and (Al-shabi and Othman, 2008). In this paper, we focus on packet switching and MINs.

Solving the problem of internal blocking in MIN's has been considered in many previous researches. Amer Arafah (1997) had developed new algorithms to resolve the internal blocking in MIN in optical domain. The developed algorithms assume that all packets of any input channel go to the same destination.

This paper is structured in the following way. In Section 2, we give a general background about optical omega networks. In Section 3, we study the performance of our new algorithm for optical omega network with the centralized buffering concept. In Section 4, we study the performance of our new algorithm for optical omega networks with wavelength conversion concept. In Section 5, we show the results obtained and compare the performance of all different algorithms. The paper is concluded in Section 6.

2. Optical omega networks

2.1. Architecture of omega network

Omega network is a type of MIN, in which, $a = b = 2$. Omega network has been chosen in this research to verify the introduced algorithm. The strength of the omega network comes from its simple structure in which links permutation is the same for all stages.

An omega network consists of stages of a number of 2×2 switching elements (SEs). In an $N \times N$ omega network, using 2×2 SE, only $\log_2 N$ stages are required to achieve full access capability. The general architecture of an omega network consists of a few stages of a number of 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 (Amer Arafah, 1997). So, if the binary representation of an input link is:

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

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

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

For example, for $n = 3$, the following matrix shows the perfect shuffle connections:

$$\sigma = \begin{bmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 0 & 2 & 4 & 6 & 1 & 3 & 5 & 7 \end{bmatrix}$$

The perfect shuffle connection of omega networks simply divides the N-channels into two halves, which are then interleaved perfectly. Basically, the 2×2 switching element (SE) has two configurations, straight or exchange as illustrated in Fig. 1

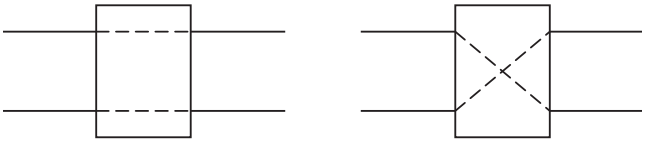

Figure 1 Configurations of a switching element.

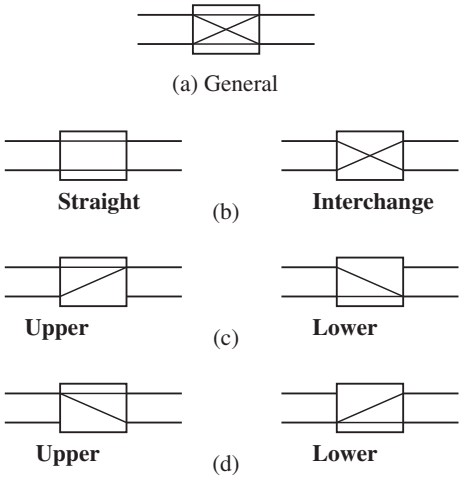

Figure 3 The general switch is illustrated in (a); the other configurations are illustrated in (b), (c) and (d).

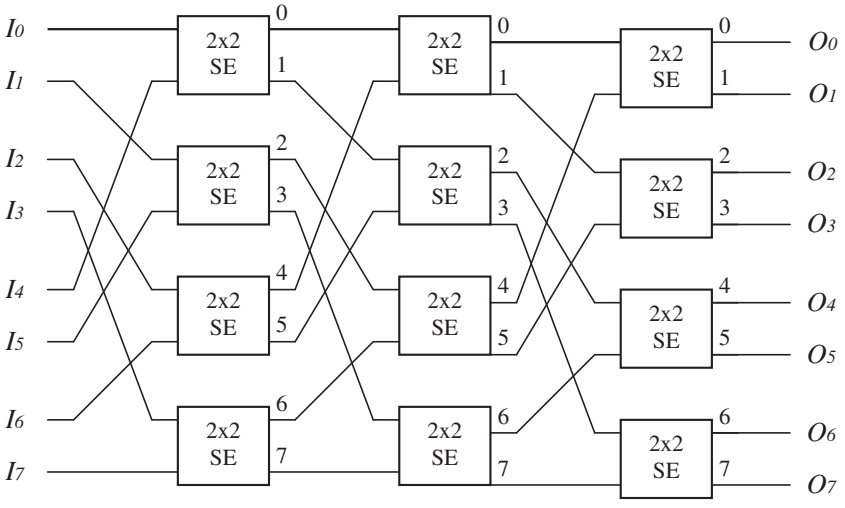

Figure 2 An 8×8 omega network, Consisting of 3-stages, with each stage consisting of a perfect shuffle followed by 4 switches.

In general, omega network consists of $n = \log_2 N$ identical stages, and each stage consists of a perfect shuffle connection followed by $N/2$ 2×2 switch elements. Fig. 2 shows an omega network with $N = 8$.

2.2. Architecture of optical omega network

An optical omega network is an omega network based on Wavelength Division Multiplexing (WDM). In an optical omega network with wavelengths $= \omega$, each input link can carry at most ω packets, each with a different wavelength from the set of available wavelengths $\Lambda = (\lambda_1, \lambda_1, \ldots, \lambda_{\omega-1}, \lambda_\omega)$. The two inputs of a 2×2 switch element can be merged and forward to either the upper or lower output link when the sets of wavelengths on both input links are disjoint. Therefore, two additional configurations have to be considered, namely, *upper merger* and *lower merger* as illustrated in Fig. 3c. Moreover, we need two additional configurations, namely, the *upper splitter* and the *lower splitter* as illustrated in Fig. 3d (Amer Arafah, 1997).

3. New algorithms to resolve internal blocking in optical omega network with centralized buffering

Considering the omega networks based on WDM which have been discussed in the previous section and illustrated in Fig. 4. In the none-buffered optical omega network, internal blocking occurs in a switch between two input packets when the two packets use the same wavelength and attempt to go to the same output links (i.e., both attempt to go to the upper output or both attempt to go to the lower output of the switch). When an internal blocking occurs, one of the two contending packets will be dropped. To reduce the dropping (and hence, increase the performance) in omega network, buffering is used. In omega networks with buffering, instead of dropping one of the two contending packets, the central controller checks if there is an available space in the buffer, if so, one of the two contending packets (chosen randomly) will be buffered. If the buffer is full, the packet will be dropped.

For example, consider the SE shown in Fig. 5, the upper link carries two packets of the wavelengths $\{\lambda_0, \lambda_1\}$; the lower link carries two packets of the wavelengths $\{\lambda_0, \lambda_2\}$ and both sets of packets attempting to go the upper link.

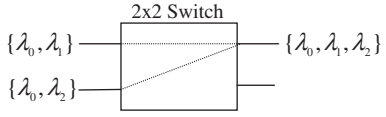

Figure 5 Internal blocking in an optical SE, one of the λ_0 input packets will be blocked.

As shown above in Fig. 5, there is a contention because the two input packets, which use λ_0 wavelength, attempt to go to the same output link (upper output link). Note that the two packets λ_1 and λ_2 do not cause internal blocking although they destined to the same output link, that is because they are using different wavelengths.

3.1. The analytical model

We first analyze the internal blocking in omega networks without buffering or wavelength conversion. The arrival rate is assumed to be Bernoulli and the packet destination is uniform. We will compute the blocking rate in omega network, i.e., given a packet entered the omega network, we will compute the probability that this packet will be internally blocked.

In addition to the definitions in the previous section, we define the following:

p_i: Mean arrival rate at stage i (at first stage, $p_1 = p$).
pd_i: Probability of blocking an input packet at stage i, given that the packet entered the network.

Given that a packet entered an n-stage omega network with mean arrival rate of p, the probability that this packet will be blocked at stage 1 is given by:

$$pd_1 = 0.25 \cdot p_1, \tag{1}$$

where p_1 is the arrival rate at stage 1 which is equal to p.

The probability that a packet will be blocked at stage 2 is given by:

$$pd_2 = 0.25 \cdot (1 - pd_1) \cdot p_2 \tag{2}$$

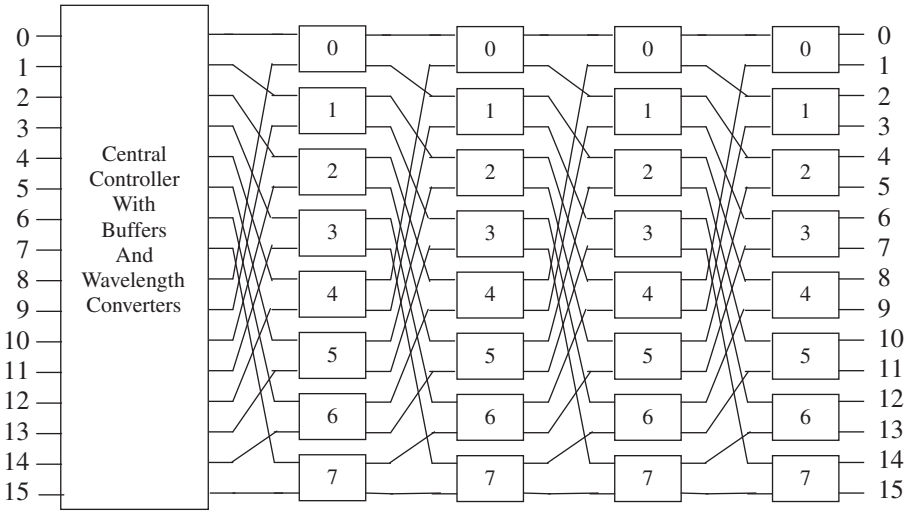

Figure 4 Block diagram for a 16×16 omega network structure with central controller and central buffer.

Similarly,

$$pd_3 = 0.25 \cdot (1 - pd_1) \cdot (1 - pd_2) \cdot p_3 \quad (3)$$

$$\vdots$$

$$pd_n = 0.25 \cdot (1 - pd_1) \cdot (1 - pd_2) \cdots (1 - pd_{n-1}) \cdot p_n \quad (4)$$

The mean arrival rate at stage i is equal to the value of the mean arrival rate of the previous stage minus the blocking probability, so:

$$p_i = p_{i-1} \cdot (1 - 0.25 \cdot p_{i-1}) \quad (5)$$

The probability of blocking a packet entered to the network is the summation of the probabilities of blocking that packet in each stage, so:

$$\text{Blocking probability} = pd = \sum_{i=1}^{n} pd_i \quad (6)$$

From the previous analysis, we computed pd, which is the probability that a packet entered to the omega network will be internally blocked. Now, we will compute the following three network values:

r_b: Buffering rate, the rate of buffered inputs to the total network inputs during one cycle.
r_d: Dropping rate, the rate of dropped inputs to the total network inputs during one cycle.
Delay: Average number of network cycles needed for a packet to reach to its destination.

Before we can compute the buffering rate, dropping rate and average packet delay, the following probabilities must be computed:

$P(Blocked = k)$: Probability that k packets will be internally blocked during a cycle.
$P(Arrivals = j)$: Probability that number of arrival packets during a cycle is j.
$P(Blocked = k/Arrivals = j)$: Probability that k packets will be internally blocked given that there are j arrivals.
$P(Buffred = i)$: Probability that i packets have been buffered during one cycle.
$P(Dropped = i)$: Probability that i packets have been dropped during one cycle.
$E\{N\}$: Mean number of arrivals during one cycle.
$E\{Buffered\}$: Mean number of buffered packets during one cycle.
$E\{Dropped\}$: Mean number of dropped packets during one cycle.

From the theorem of total probability:

$$P(Blocked = k) = \sum_{j=k}^{N} P(Arrivals = j) \cdot P(N_{block} = k/Arrivals = j) \quad (7)$$

Since arrival is Bernoulli, $P(Arrivals = j)$ is given by:

$$P(Arrivals = j) = \binom{N}{J} \cdot p^j \cdot (1 - p)^{N-j} \quad (8)$$

Using pd computed in (1), we can compute $P(Blocked = k/Arrivals = j)$ as follows:

$$P(Blocked = k/Arrivals = j) = \binom{j}{k} \cdot pd^k \cdot (1 - pd)^{j-k} \quad (9)$$

From (7)–(9):

$$P(N_{block} = k) = \sum_{j=k}^{N} \binom{N}{j} \cdot p^j \cdot (1 - p)^{N-j} \cdot \binom{j}{k} \cdot pd^k \cdot (1 - pd)^{j-k} \quad (10)$$

Also,

$$P(Buffered = 0) = P(Blocked = 0)$$
$$P(Buffered = 1) = P(Blocked = 1)$$
$$\cdots$$
$$\cdots$$
$$P(Buffered = BufSize - 1) = P(Blocked = BufSize - 1)$$
$$P(Buffered = BufSize) = \sum_{k=BufSize}^{N} P(Blocked = k)$$

So, we get:

$$P(Buffered = j) = \begin{cases} P(Blocked = j), & 0 \leqslant j < BufSize \\ \sum_{k=BufSize}^{N} P(Blocked = k), & j = BufSize \\ 0, & j > BufSize \end{cases} \quad (11)$$

$$E\{Buffered\} = \sum_{j=1}^{BufSize} j \cdot P(Buffered = j)$$

Since the input arrivals are independent, $E\{Arrivals\}$ is simply equal to $p * N$. So,

$$\text{Buffering rate} = r_b = \frac{E\{Buffered\}}{E\{Arrivals\}} = \frac{E\{Buffered\}}{p * N}$$

$$\text{Delay} = \sum_{j=1}^{\infty} j \cdot r_b^{j-1} \cdot (1 - r_b)$$

Similarly,

$$P(Dropped = j) = \begin{cases} \sum_{k=0}^{BufSize} P(Blocked = k), & j = 0 \\ P(Blocked = BufSize + j), & 0 < j \leqslant N - BufSize \\ 0, & j > N - BufSize \end{cases} \quad (12)$$

So,

$$E\{Dropped\} = \sum_{j=1}^{BufSize} j \cdot P(Dropped = j)$$

$$\text{Dropping rate} = r_d = \frac{E\{Dropped\}}{E\{Arrivals\}} = \frac{E\{Dropped\}}{p * N}$$

3.1.1. Computing r_b, r_d and delay

The computation of r_b depends of the arrival rate (p), which has two sources:

(a) External arrival rate (λ), which is the network arrival rate.
(b) Retransmitted packets which are buffered in the previous cycle, which depends on the buffering rate (r_b) and equal to $r_b \cdot p$

So, the actual arrival rate is given by:

$$p = \lambda + (1 - \lambda) \cdot r_b \cdot p$$

$$\text{or,} \quad p = \frac{\lambda}{1 - r_b + r_b \cdot \lambda}$$

The computation of buffering rate (r_b), dropping rate (r_d) and delay for a given external arrival rate (λ) is done through the following iterations:

(1) Set the active actual rate $p = \lambda$ as an initial value.
(2) For the current p, compute $P(N_{block} = k)$, $P(N_{buf} = j)$, $E\{N_{buf}\}$ and then r_b.
(3) Using r_b, compute the new arrival rate using the equation: $np = \frac{\lambda}{1-r_b+\lambda \cdot r_b}$.
(4) If $abs(np - p) < error$ then stop iterating, otherwise set $p = np$ and go to step 2.

Using the set of packets for each link and the given permutation in addition to the knowledge of the behavior of omega networks, the central controller will execute the following algorithm to identify all packets that will cause internal blocking, and will buffer them before sending all the other packets through the network. Therefore, we resolve the problem of internal blocking.

3.2. The new algorithm

The algorithm processes each input pattern in one cycle. In an 16×16 omega network, each cycle consists of routing the input pattern through four stages. Since we assumed that the buffered packets have priority over the incoming new packets, the central buffer checks all buffered packets at the beginning of each new cycle, if a new input used the same input link, the new input will be simply ignored.

Optical omega network with centralized buffering algorithm

```
inputs = get input pattern from the random input file
for all packets in the buffer from previous cycle, do the following
– remove a packet from the buffer
– Attach the packet to it's source, if a new packet coming from that
source, then suspend new incoming packet.
for stage = 1 to n do
  for switch = 0 to N div 2–1 do
    for j = 0 to ω − 1 do
        pkt(0) = upper input packet that uses wavelength (j)
        pkt(1) = lower input packet that uses wavelength (j)
      if pkt(0) is idle then p0 = −1
        else if pkt(0) goes to upper link then p0 = 0
          else p0 = 1;
      if pkt(1) is idle then p1 = −1
        else if pkt(1) goes to upper link then p1 = 0
            else p1 = 1;
      ** Routing **
      if (p0 = −1) and (p1 = −1) then
        do nothing; {both inputs are idle}
    else if (P0 = −1) or (P1 = −1) then
          pass the active input {One idle input}
      else if (P0 < > P1) then
            pass the two inputs {Straight or Interchange}
        else
            {Internal blocking state}
          R = Random integer number in the range [0, 1];
            pass pkt(R);
      if (Buffer.Size < BufSize) then
            Add pkt(1 − R) to the buffer;
            TotBuf = TotBuf + 1;
      else
        TotDrop = TotDrop + 1;
```

Although this algorithm will improve the performance of omega networks based on WDM, the improvement achieved is not as great as that achieved by introducing wavelength converters in the network architecture.

4. New algorithms for optical omega network with wavelength conversion

In this section, we use the same architecture of an omega network which has been discussed in the last section. However, we include Wavelength Converters in the central controller of that network as illustrated in Fig. 4. The purpose of a wavelength converter is to convert the wavelength λi of a packet to another wavelength λj, where $\lambda j \neq \lambda i$. Therefore, if we have internal blocking, and we manage to convert the wavelengths of packets that cause the internal blocking, we can reduce number of packets to be buffered by the central controller, and the performance of that network will be improved. Note that a packet can have at most one wavelength conversion and this happens inside the central controller. The number of packets which can have their wavelengths converted is limited by the number of available converters. Packets which will cause internal collision but which cannot be wavelength converted are buffered. If we run out of buffers, packets are dropped.

In the optical omega network each link is divided into ω divisions, so it can carry at most ω packets, each of which has a unique wavelength in the range $(0, \omega - 1)$. In the normal cases, not all of the ω slots are utilized; this depends on the input arrival rate. This algorithm utilizes the unused wavelengths in input links. Depending on the way that unused wavelengths are utilized, two algorithms are introduced. One utilizes only wavelengths that are unused in both input links. The other algorithm makes an optimum utilization of unused wavelengths since it utilizes wavelengths that are unused in any input link.

In both algorithms, the wavelength conversion is done in the central controller. In the beginning of each cycle, the central controller processes input packets and resolves internal blocking (by wavelength conversion or dropping) and then passes the packets to the omega network to be routed without conflicts. The central controller processes the input packets by doing virtual routing (i.e., simulating the routing done in the omega network) to detect packets conflicts. Advantages of using centralized wavelength conversion over distributed wavelength conversion are:

Optimum usage of available wavelength converters. In the distributed wavelength conversion, the available wavelength converters are distributed over SE's. In this case, it happens that packets are dropped in a SE because all available wavelength converters are used while wavelength converters in other SE's are not fully used. In the other hand, this situation does not happen in centralized wavelength conversion because available wavelength converters are shared among all SE's.

In the distributed wavelength conversion, it happens that a packet is converted more than once which consumes the available wavelength converters, but that does not happen in the centralized wavelength converters.

In the centralized wavelength conversion, SE's are simple since there are no wavelength converters or controllers in them.

4.1. Algorithm 1: free wavelengths must be unused in both input links

To illustrate the wavelength conversion concept, consider the 2×2 switch shown in Fig. 6. In this example we assume the following:

$\omega = 4$, set of available wavelengths is $\{\lambda_0, \lambda_1, \lambda_2, \lambda_3\}$
Both inputs attempts to go to the upper link.
Upper link inputs = $\{\lambda_0, \lambda_1\}$
Lower link inputs = $\{\lambda_0, \lambda_2\}$

As shown above in Fig. 6, there is a contention because the two input packets which use λ_0 wavelength attempt to go to the same output link (upper output link). To resolve the contention, one of the two packets has been dropped.

If wavelength conversion is implemented, instead of dropping one of the λ_0 packets to resolve the contention, its wavelength is converted to one of the unused wavelengths, which is λ_3 as in Fig. 7. After wavelength conversion, the packets can be routed to the upper link without contention. So, we expect that using wavelength conversion decreases the dropping probability and hence increasing the network performance.

The wavelength conversion is done in the central controller. If two packets attempting to use the same output link, the controller try to convert the wavelength of one of the two packets (chosen randomly). If there is available wavelength converters and free wavelength, the central controller will convert the wavelength of one of the contending packets to a new wavelength chosen randomly from the set of the unused wavelengths. After all possible wavelengths conversions and necessary droppings are done, the central controller passes the input packets to the omega network and the packets will be routed through it without internal blocking.

Since the wavelength conversion is done in the central controller before passing input packets to the omega network, the set of available wavelengths (S) for a packet in a switching element (SE) at a specific stage (i) is the intersection of available wavelengths in SE with the sets of unused wavelengths along the path of the packet. To illustrate how the set of available wavelengths for a specific packet is computed, consider the example shown in Fig. 8. In this example a packet entering from input link 0 and its destination output is 0. S_i and O_i are sets of unused wavelengths in the specified links. The set of available wavelengths at stage 1 is $S_1 \cap O_1$, at stage 2 available wavelengths set is $S_1 \cap S_2 \cap O_2$, at stage 3 the set is given by $S_1 \cap S_2 \cap S_3 \cap O_3$ and at stage 4 the set is $S_1 \cap S_2 \cap S_3 \cap S_4 \cap O_4$.

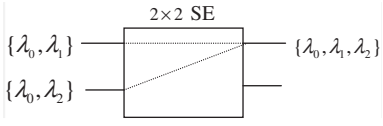

Figure 6 No wavelength conversion used. One of the λ_0 packets has been dropped.

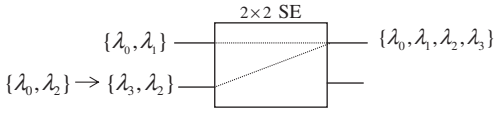

Figure 7 Wavelength conversion is used. Wavelength of λ_0 packet (in one of the two inputs) has been changed to λ_3.

Wavelength conversion Algorithm 1

```
inputs = get input pattern from the random input file
CurWC_No = WCSize;
  for stage = 1 to n do
    for switch = 0 to N div 2–1 do
        FreeWaves = Set of wavelengths that are not used in both
switch inputs;
       for j = 0 to ω – 1 do   ** Route switch input packets **
           pkt(0) = upper input packet that uses wavelength (j)
           pkt(1) = lower input packet that uses wavelength (j)
         if pkt(0) is idle then p0 = –1
           else if pkt(0) goes to upper link then p0 = 0
                  else p0 = 1;
         if pkt(1) is idle then p1 = –1
           else if pkt(1) goes to upper link then p1 = 0
                  else p1 = 1;
      ** Routing **
    if (p0 = –1) and (p1 = –1) then
          do nothing; ** both inputs are idle **
    else if (P0 = –1) or (P1 = –1) then
            pass the active input **One idle input **
          else if (P0 < > P1) then
              pass the two inputs ** Straight or interchange **
      else {Internal blocking state}
        R = Random integer number in the range [0, 1];
          pass pkt(R);
        for j = stage-1 down to 1 do
FreeWaves = FreeWaves ∩ Sj
        ** Where Sj is the set of unused wavelengths in
          the link that carries pkt(1 – R) at stage j **
        if (CurWC_No > 0) and (FreeWaves.Size > 0) then
          convert pkt(1 – R) wavelength to new wavelength
chosen
          randomly from FreeWaves;
        if pkt(1 – R) is not converted before,decrement
CurWC_No;
          delete the chosen wavelength from FreeWaves
          TotConv = TotConv + 1;
          For j = stage-1 down to 1 update Sj
        Else
            TotDrop = TotDrop + 1; ** Drop the blocked input
**
```

4.2. Algorithm 2: free wavelengths are the unused ones in any input link

In this algorithm, any wavelength that is not used in an input link can be used in the wavelength conversion given that the wavelength conversion resolves the internal blocking. At the beginning of processing the two inputs of a switch, sets of unused wavelength are generated by the central controller. There are five sets associated with each switch to indicate the unused wavelengths, two for the upper input link, two for the lower input link and the fifth is common for the two inputs. The five sets are as follows:

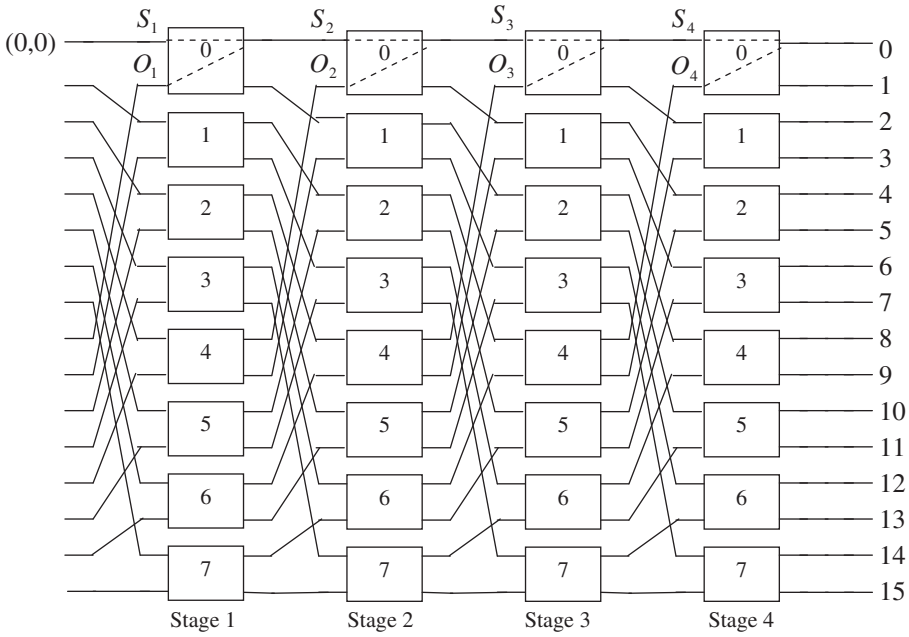

Figure 8 Example illustrates how the available wavelength set for a packet in a specific stage is computed.

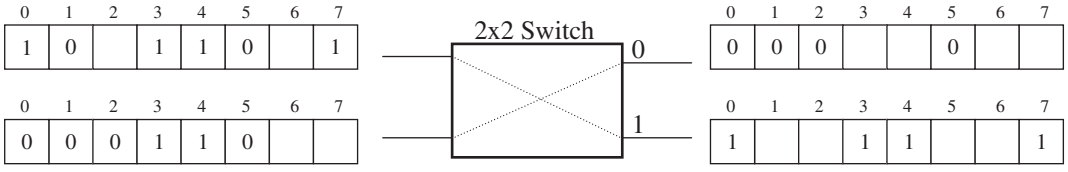

Figure 9 Case1: Routing inputs without wavelength conversion, 4 packets have been dropped.

Set(0, 0): Unused wavelengths in upper link that can accommodate a packet destined to the output upper link.
Set(0, 1): Unused wavelengths in upper link that can accommodate a packet destined to the output lower link.
Set(1, 0): Unused wavelengths in lower link that can accommodate a packet destined to the output upper link.
Set(1, 1): Unused wavelengths in lower link that can accommodate a packet destined to the output lower link.

Common Set: Wavelengths that are not used in the upper link neither the lower link. So, a wavelength from this set can accommodates a packet destined to upper or lower output link.

To illustrate how the unused wavelengths sets are generated and utilized, consider the following example. A 2×2 general switch shown in Fig. 9 has maximum wavelengths of $\omega = 8$. In the figure, the number in each slot indicates the destination of the packet and the number above the slot indicates the corresponding wavelength.

There are three cases: first case is shown in Fig. 9, in which wavelength conversion is not implemented, in this case, 4 packets have been dropped. The dropped packets have the wavelengths $\{\lambda_1, \lambda_3, \lambda_4, \lambda_5\}$.

Second case is shown in Fig. 10. In this case, the wavelengths that are not used in either input have been utilized (Algorithm 1). In this example, only $\{\lambda_6\}$ is not used by both inputs. The conflict in λ_6 has been resolved by converting one of the two inputs wavelength from λ_1 to λ_6 and then routing them without conflict. In this case, only 3 packets have been dropped.

The last case shown in Fig. 11 below illustrates Algorithm 2, in which any unused input wavelength may be utilized in wavelength conversion to resolve input contentions in the wavelengths $\{\lambda_1, \lambda_3, \lambda_4, \lambda_5\}$. In the upper input link, $\{\lambda_2, \lambda_6\}$ are not used. In the lower input link, $\{\lambda_6, \lambda_7\}$ are not used. The unused wavelengths sets are:

$$\text{Set}(0, 1) = \{\lambda_2, \lambda_6\}$$
$$\text{Set}(1, 0) = \{\lambda_7\}$$
$$\text{Common Set} = \{\lambda_2, \lambda_6\}$$
$$\text{Set}(0, 0) = \text{Set}(1, 1) = \phi$$

In this particular example, wavelength conversions in the upper input link are $(\lambda_1 \rightarrow \lambda_6)$ and $(\lambda_3 \rightarrow \lambda_1)$, and in the lower input link: $(\lambda_4 \rightarrow \lambda_6)$ and $(\lambda_5 \rightarrow \lambda_7)$.

In general, when a contention occurs and the wavelength in an input link have to be changed, the algorithm searches first in the two sets corresponding to the input link, if no empty wavelength found, the algorithm searches the common set. Since wavelength elements in the common set are always can be used in wavelength conversions at both inputs, we made the search in this set as the last choice.

As in Algorithm 1, since the wavelength conversion is done in the central controller before passing input packets to the omega network, the set of available wavelengths (S) for a packet in a switching element (SE) at a specific stage (i) is the intersection of available wavelengths in SE with the sets of unused wavelengths in links among the path of the packet.

Wavelength conversion Algorithm 2

```
inputs = get input pattern from the random input file
CurWC_No = WCSize; * Initialize current available converters *
for stage = 1 to n do
  for switch = 0 to N div 2–1 do
  ** generate the set for unused wavelengths **
  generate FreeWaves Sets ** sub-algorithm **
  for j = 0 to ω − 1 do
    pkt(0) = upper input packet that uses wavelength (j)
    pkt(1) = lower input packet that uses wavelength (j)
    if pkt(0) is idle then p0 = −1
    else if pkt(0) goes to upper link then p0 = 0 else p0 = 1;
    if pkt(1) is idle then p1 = −1
    else if pkt(1) goes to upper link then p1 = 0 else p1 = 1;
          ** Routing **
    if (p0 = −1) and (p1 = −1) then do nothing;
    else if (P0 = −1) or (P1 = −1) then pass the active input
    else if (P0 < > P1) then pass the two inputs *straight or exchange*
    else *Internal blocking state*
    R = Random integer number in the range [0, 1];
    Path_Sets = {all waves} ** Initial value **
    for j = stage-1 down to 1 do Path_Sets = Path_Sets ∩ Sj
    ** Where Sj is the set of unused wavelengths in the link that carries pkt(1-R) at stage j **
    if (FreeWaves(1 − R, P0) ∩ Path_Sets) is not empty and (CurWC_No > 0) then
      pass pkt(R);
        convert pkt(1 − R) using the set FreeWaves(1 − R, P0) ∩ Path_Sets;
    else if (FreeWaves.Size ∩ Path_Sets) is not empty
        and (CurWC_No > 0) then
    pass pkt(R);
      convert pkt(1 − R) using the set FreeWaves ∩ Path_Sets;
    else
      pass pkt(R);
      TotDrop = TotDrop + 1; ** Drop the blocked input **
```

Sub-algorithm for generating sets for unused wavelengths

```
FreeWaves(0, 0).Size = 0; * Initialize Set(0, 0)
FreeWaves(0, 1).Size = 0; * Initialize Set(0, 1)
FreeWaves(1, 0).Size = 0; * Initialize Set(1, 0)
FreeWaves(1, 1).Size = 0; * Initialize Set(1, 1)
FreeWaves.Size = 0; * Initialize Common Set
for j = 0 to ω − 1 do
  If wavelength (j) is not used in both inputs then
    FreeWave.Set = FreeWave.Set + {j}
      FreeWave.Size = FreeWave.Size + 1;
  else if wavelength (j) is not used in upper input then
        if lower input packet using wavelength (j) destined to lower
          output link then
            FreeWaves(0, 0).Set = FreeWaves(0, 0).Set + {j}
                FreeWaves(0, 0).Size = FreeWaves(0, 0).Size + 1;
          else
            FreeWaves(0, 1).Set = FreeWaves(0, 1).Set + {j}
            FreeWaves(0, 1).Size = FreeWaves(0,1).Size + 1;
      else
      if upper input packet using wavelength (j) destined to lower
          output link then
          FreeWaves(1, 0).Set = FreeWaves(1, 0).Set + {j}
            FreeWaves(1, 0).Size = FreeWaves(1, 0).Size + 1;
      else
          FreeWaves(1, 1).Set = FreeWaves(1, 1).Set + {j}
            FreeWaves(1, 1).Size = FreeWaves(1, 1).Size + 1;
```

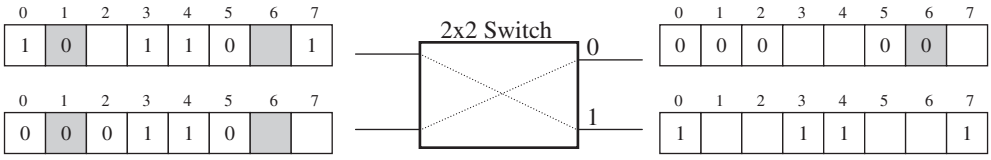

Figure 10 Case2: Using wavelength conversion. Utilizing unused wavelengths in both inputs (w6), 3 packets have been dropped.

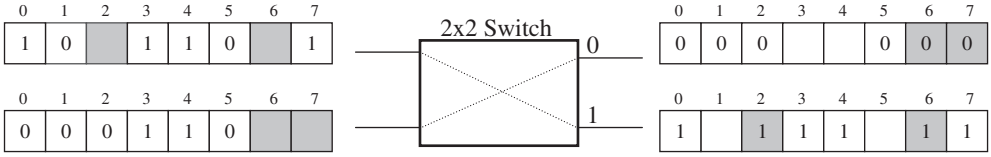

Figure 11 Case3: Using wavelength conversion. Utilizing all unused wavelengths (w2, w6 and w7). Dropping = 0.

In optical omega network with centralized buffering and wavelength conversion, when internal blocking occurs, we try to make a wavelength conversion for one of the two contending packets, if wavelength conversion is not possible, one of the two contending packets will be buffered.

In this algorithm, we give priority for wavelength conversion over buffering since wavelength conversion process will not affect packet delay as buffering does.

5. Performance results

A simulation program has been implemented to measure the performance of the three algorithms introduced in this paper. For each algorithm, the corresponding 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 2500 random input connections. Furthermore, the previous process has been repeated 10 times to compute the confidence interval. The averages and confidence intervals have been computed according to the following equations:

$$\Pr\left(\overline{X_n} - \frac{c\cdot\sigma'}{\sqrt{n}} \leqslant \mu \leqslant \overline{X_n} + \frac{c\cdot\sigma'}{\sqrt{n}}\right) = 0.95$$

$$\text{where } c = 1.833,\ \sigma' = \sqrt{\frac{S_n^2}{n-1}},\ \text{and } S_n^2 = \sum_{i=1}^{n}(X_i - \bar{X}_n)^2$$

Simulation results are presented graphically (Fig. 12 through Fig. 17). Before we start discussing simulation results, we list some definitions of the performance parameters that we will use.

5.1. Dropping rate

The rate of internally dropping an input packet. It is computed by dividing the total internally dropped packets by the total input packets entering the network.

$$\text{Dropping rate} = \frac{\text{Total dropped packets}}{\text{Total input packets}}$$

5.2. Buffering rate

The rate of buffering an input packet. It is computing by dividing the total buffered packets by the total input packets entering the network.

$$\text{Buffering rate} = \frac{\text{Total buffered packets}}{\text{Total input packets}}$$

5.3. Wavelength conversion rate

The rate of doing a wavelength conversion for an input packet. It is computed by dividing the total wavelength conversions by the total packets entering the network.

$$\text{Wavelength conversion rate} = \frac{\text{Total wavelength converted packets}}{\text{Total input packets}}$$

Figs. 12–14 show the analytical results and the simulation results for the omega network with centralized buffering. The figures show that the results obtained using analysis are very close to those obtained by simulation. Fig. 12 compares the dropping probability between the analytical and simulation results. Fig. 13 compares the buffering probability between the analytical and simulation results. Fig. 14 compares the average packet delay between the analytical and simulation results.

In the rest of this section, we discuss the simulation results for the algorithms in the previous sections, buffering, wavelength conversion and both centralized buffering and wavelength conversion.

5.4. Centralized buffering, no wavelength conversion

Fig. 15 shows that using centralized buffering decreases internal blocking significantly, and thus increasing the optical omega network performance. The figure shows that dropping rate decreases with buffer size increase, until it becomes zero at $B = 256$, since the simulated omega network inputs are 16, with $\omega = 16$.

Fig. 16 illustrates the buffering rate for different arrival rates and different buffer sizes. For the curves shown in the figure (except $B = 256$), the buffering rate start increasing almost linearly with the arrival rate up to a maximum value, after this value the curves start decreasing smoothly. The reason for curves descending, after a specific arrival rate, is that dropping starts increasing after that arrival rate because there are no available buffers for a blocked packet so it will be dropped. The curves start decreasing because there are no available buffers, that is why the maximum value shift to the right as the buffer size (B) increase. The curves shown in Fig. 16 help in

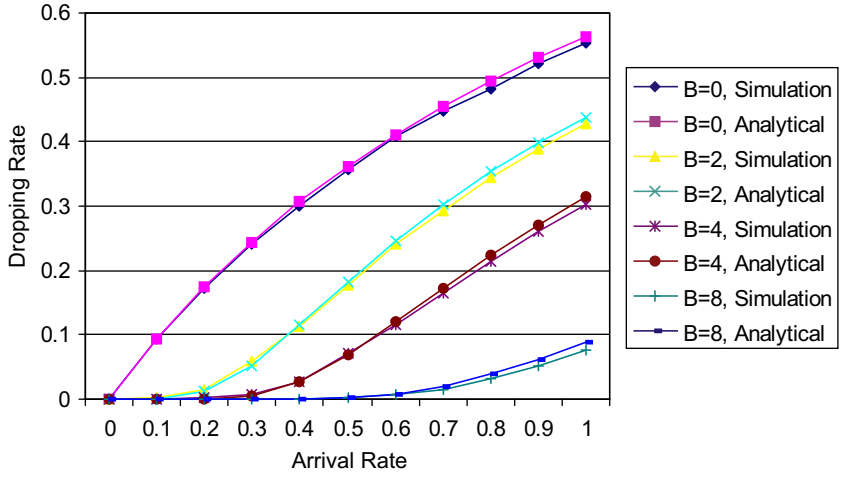

Figure 12 Dropping probability centralized buffering, comparing simulation with analytical results.

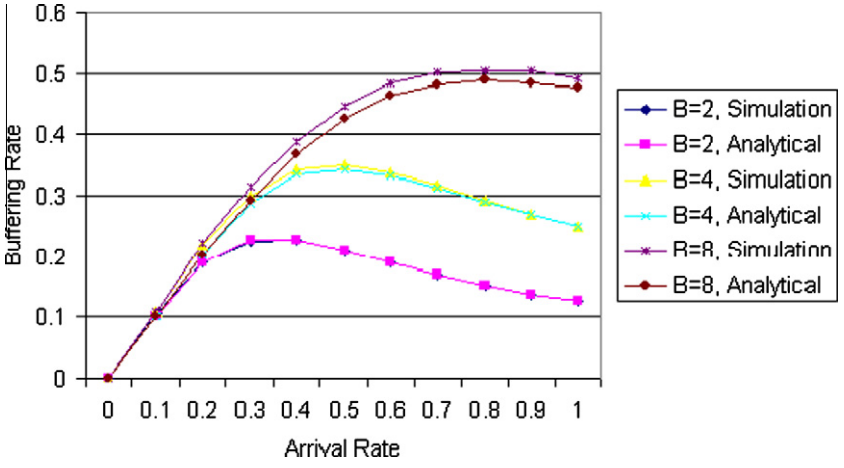

Figure 13 Buffering probability. centralized buffering, comparing simulation with analytical results.

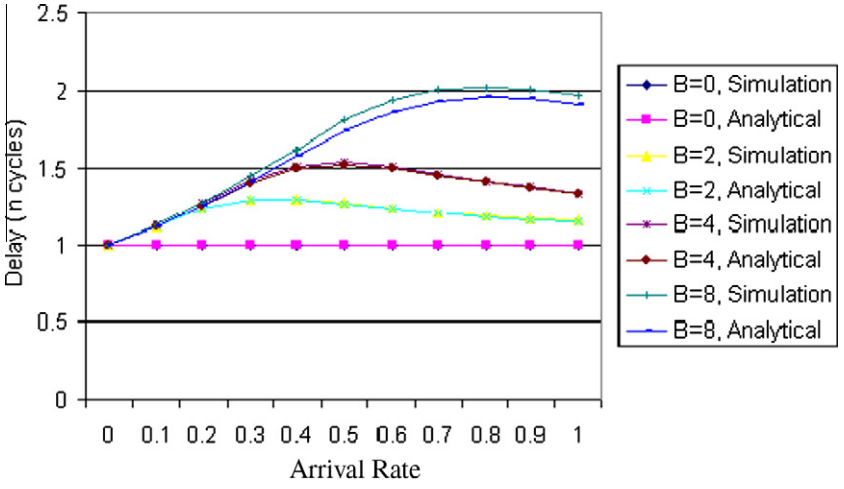

Figure 14 Average packet delay centralized buffering, comparing simulation with analytical results.

computing the optimum buffer size for a network with a specific average arrival rate. For example, for a network with average arrival rate of 0.4, the optimum buffer size is $B = 32$.

Fig. 14 shows the average packet delay vs. arrival rate for different buffer sizes. It is clear from Fig. 15 that as the buffer size increases, the dropping rate decreases, but the cost is

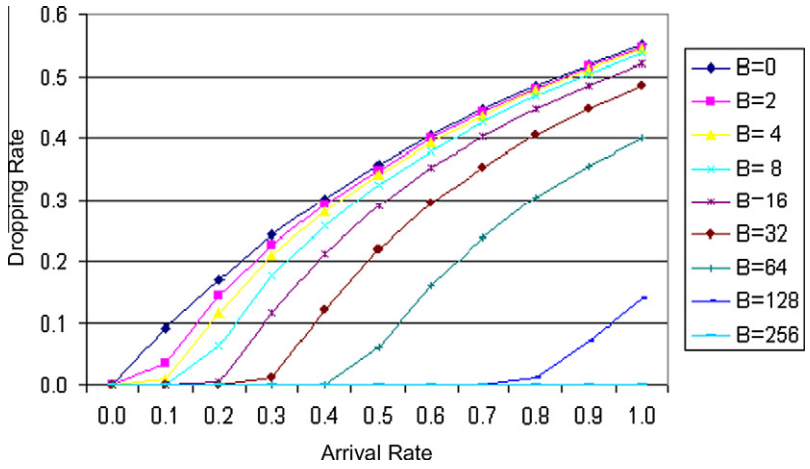

Figure 15 Dropping probability centralized buffering, no wavelength conversion.

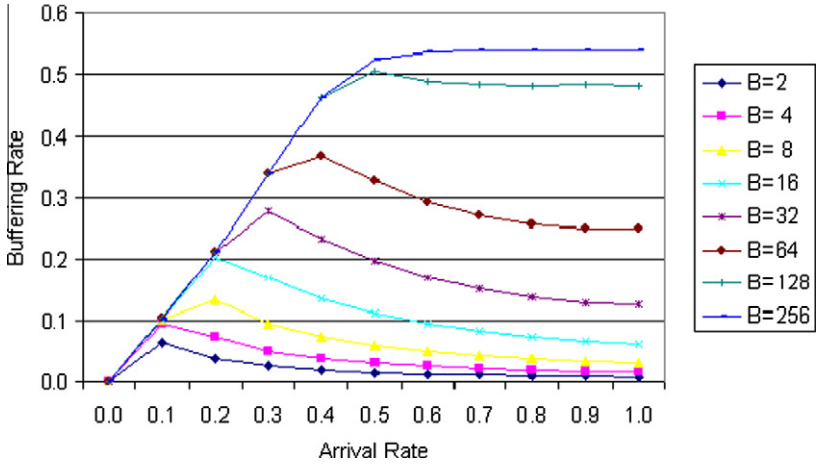

Figure 16 Buffering probability centralized buffering, no wavelength conversion.

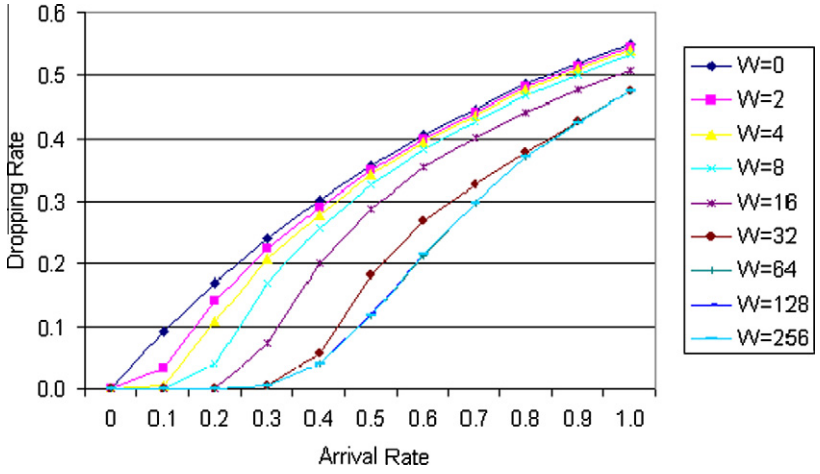

Figure 17 Dropping probability, no buffering, limited wavelength converters, wavelength conversion Algorithm 1.

increasing in average packet delay. Therefore, it is very important to take care of the average packet delay and not use extra buffers than needed especially for the application which are delay sensitive.

5.5. Wavelength conversion, no buffering

If we use wavelength converters, we can obtain a considerable improvement. Fig. 17 illustrates the packet dropping probabil-

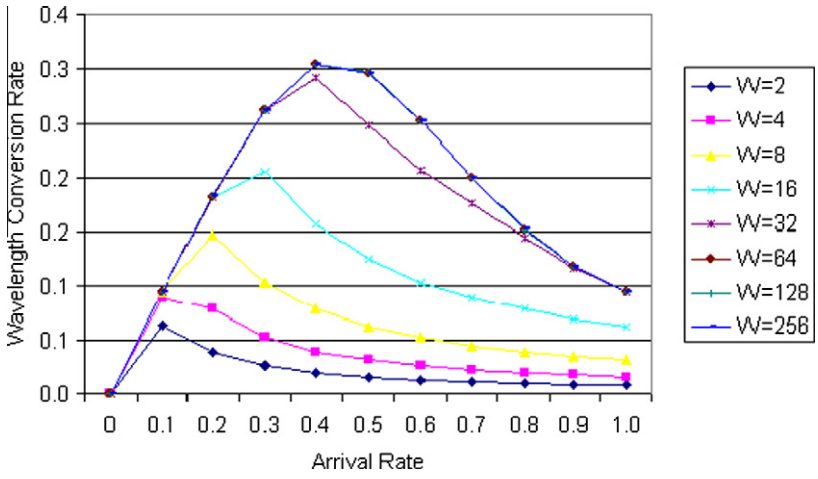

Figure 18 Wavelength conversion probability, no buffering, limited wavelength converters, wavelength conversion Algorithm 1.

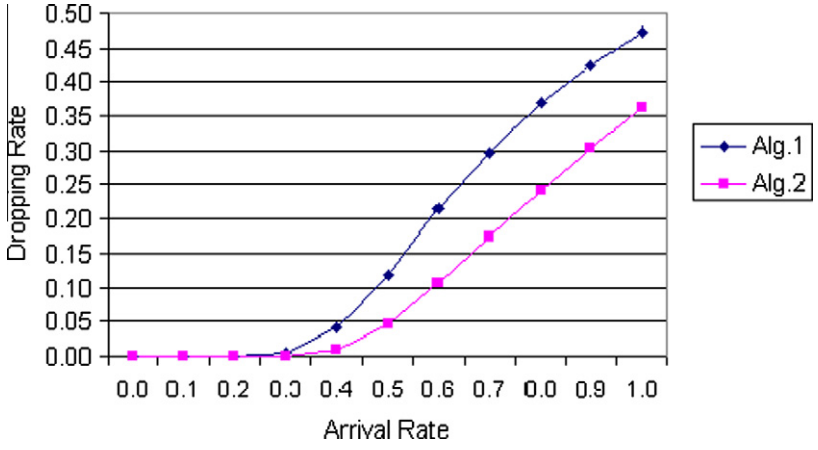

Figure 19 Evaluating enhancement of using Algorithm 2 over using Algorithm 1, dropping probability is used as the evaluation key no buffering, $W = 128$.

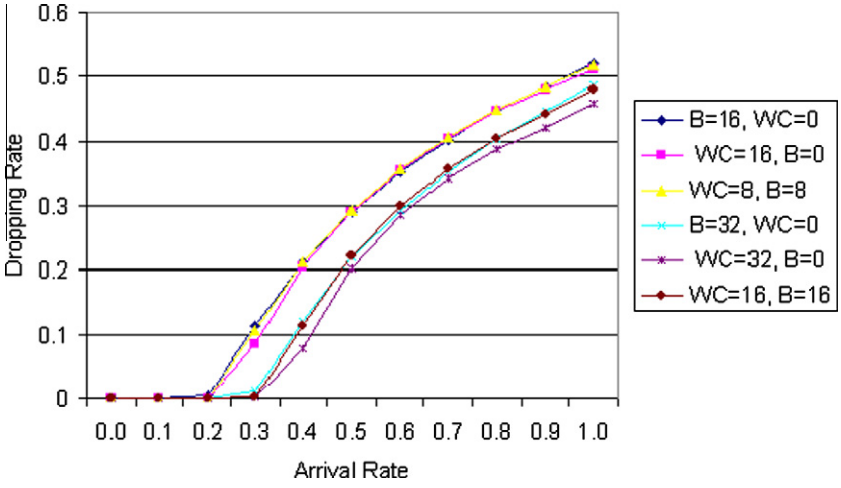

Figure 20 Dropping probability for different omega networks.

ity versus arrival rate. It shows that using just four wavelength converters will improve the system performance. As expected, the packet dropping probability decreases with increasing number of wavelength converters. Note that when the arrival rate is equal one, the packet dropping probability will have no improvement over a network without wavelength convert-

ers, since all wavelengths are utilized and none of them are available for conversion.

Figs. 17 and 18 show the dropping and wavelength conversion probabilities resulting from using Algorithm 1 wavelength conversion. Each curve corresponds to a maximum number of available wavelength converters (*WC_Size*). Fig. 18 indicates that the wavelength conversion rate starts increasing up to a maximum value and then start decreasing at a specific arrival rate. The reason of wavelength decreasing is that after a specific arrival rate (depends on *WC_Size*), the wavelength conversion is not possible because of one of two reasons:

There are no available free wavelength converts (this often occurs when *WC_Size* is small).

There are no available unused wavelengths to be used in the conversion (this often occurs when *WC_Size* is large).

Algorithm 2 gives a good improvement over Algorithm 1, especially when W is large, because the effect of the second reason above has been decreased. Fig. 19 shows the improvement of using Algorithm 2 over using Algorithm 1 for input packets that have different destinations.

5.6. Centralized buffering and wavelength conversion

Fig. 20 shows the dropping and buffering probabilities for an optical omega network with different buffer sizes and number of wavelength converters. It shows that the packet dropping probability for a system with 8 buffers and 8 wavelength converters is the same as a system with 16 buffers or a system with 16 wavelength converters.

6. Conclusion

In this research we introduce several algorithms to resolve the problem of internal blocking in WDM omega network. The concept of these algorithms is based on a central controller which acts as an interface in front of the network. Once the central controller resolves the internal blocking by buffering, wavelength conversion, or dropping, it directs the packets through the network without any collision.

We analyze the performance of the omega networks with modification to a 2×2 switch element to incorporate the new configurations, namely the splitters and the mergers. We have found that the structure of the central control unit is very simple to implement with minimal buffers available only at the central controller rather than being distributed over all 2×2 switch elements. Therefore, we consider centralized buffers as a basis of our research to increase the utilization of the buffers. In addition, we add wavelength converters in the central controller. Rather than buffering a packet in the current switching cycle, we convert its wavelength to another wavelength to enable the omega network to accommodate it. This has increased the rate of transmission and increased the utilization of the network.

In our algorithms we consider more realistic assumptions for the central controller. We assume that the permutations between a set of inputs and a set of outputs are random, and in addition, the buffered packets in the present switching cycle will be considered with newly arrived packets in the next cycle. Using these assumptions, we analyze the performance of the omega network. We found a considerable improvement in the network performance using centralized buffers and wavelength converters.

References

Al-Shabi, M.A., Othman, M., 2008. A new algorithm for routing and scheduling in optical omega network. International Journal of The Computer, The Internet and Management 16 (1), 26–31.

Mohammed Amer Arafah. 1997. Centralized Buffering and wavelength Conversion in Multistage Interconnection Networks, A Ph.D. Dissertation, University of Southern California, December 1997.

Katangur, Ajay K., Akkaladevi, Somasheker, Pan, Yi, 2007. Analyzing the performance of optical multistage interconnection networks with limited crosstalk. Cluster Computing 10 (2), 241–250.

Kuo-Chun Lee. 1994. Wavelength Conversion and Wavelength Routing in All-Optical Networks, A Ph.D. Dissertation, Electrical Engineering, University of Southern California, August 1994.

Liboiron-Ladouceur, O., Bergman, K., 2007. Optimized switching node for optical multistage interconnection networks. IEEE Photonics Technology Letters 19, 1179–1190.

Liboiron-Ladouceur, O., Shacham, A., Small, B.A., Lee, B.G., Wang, H., Lai, C.P., Biberman, A., Bergman, K., 2008. The data vortex optical packet switched interconnection network. IEEE/OSA Journal of Lightwave Technology 26 (13), 1777–1789.

Liu, L., Yang, Y. 2008. Achieving 100% throughput in input-buffered WDM optical packet interconnects. In: Proceedings of 22nd IEEE International Parallel and Distributed Processing Symposium, Miami, FL, April 2008.

Lu, E., Zheng, S.Q., 2007. Fast reconfiguration algorithms for time, space, and wavelength dilated optical benes networks. International Journal of Parallel, Emergent and Distributed Systems 22 (1), 39–58.

Ngo, H.Q., Pan, D., Yang, Y., 2007. Optical switching networks with minimum number of limited range wavelength converters. IEEE/ACM Transactions on Networking 15 (4), 969–979.

Patel, Janak H., 1981. Performance of processor-memory interconnection for multi-processors. IEEE Transactions on Computers C-30 (Oct.).

Shares, Schow, C.L., Doany, F.E., Budd, R.A., Baks, C.W., John, R.A., Kash, J.A., Liboiron-Ladouceur, O., Dangel, R., Horst, F., Offrein, B.J., 2007. Terabus and beyond: prospects of waveguide-based optical interconnects (invited). IEEE 10th International Symposium on Contemporary Photonics Technology (CPT) 10 (E-2), 43–46.

Zhang, Z., Yang, Y., 2006. Optimal scheduling in buffered WDM interconnects with limited range wavelength conversion capability. IEEE Transactions on Computers 55 (1), 71–82.

Zhang, Z., Yang, Y., 2007. A novel analytical model for switches with shared buffer. IEEE/ACM Transactions on Networking 15 (5), 1191–1203.