Add-drop Benes network for scalable optical packet networks

Chankyun Lee and June-Koo Kevin Rhee*

Department of Electrical Engineering, Korea Advanced Institute of Science and Technology, 373-1 Guseong-dong, Yuseong-gu, Daejeon 305-701, South Korea
**rhee.jk@kaist.ac.kr*

Abstract: For the scalable optical packet transport networks, an innovative design of add-drop Benes network (ADBN) is presented where the cost and energy consumption can be considerably reduced by element savings in the architecture. In a WDM optical packet transport switching node, the ADBNs are interconnected to achieve buffer sharing among multiple ADBNs. A corresponding switch configuration algorithm and architecture rules for the single ADBN and shared ADBN are proposed to mitigate the limited connection capability of the proposed ADBN designs. Switch scalability is verified in consideration of a crosstalk noise performance and element counts.

OCIS codes: (060.1810) Buffers, couplers, routers, switches, and multiplexers; (060.4250) Networks; (060.4259) Network, packet-switched; (060.6719) Switching, packet.

References and links

1. J.-K. K. Rhee, *et al*., “Power and cost reduction by hybrid optical packet switching with shared memory buffering,” IEEE Commun. Mag. (to be published).
2. G. Shen and R. Tucker, “Energy-minimized design for IP over WDM networks,” IEEE J. Opt. Commun. Netw. **1**(1), 176–186 (2009).
3. V. W. S. Chan, G. Weichenberg, and M. Médard, “Optical flow switching,” in *Workshop on Optical Burst Switching* (IEEE, San Jose, 2006), pp. 1–12.
4. K. Nashimoto, “PLZT Waveguide devices for high speed switching and filtering,” in *Optical Fiber Communication Conference and Exposition and The National Fiber Optic Engineers Conference,* (Optical Society of America, San Diego, 2008), paper OThE4, http://www.opticsinfobase.org/abstract.cfm?URI=OFC-2008-OThE4.
5. V. E. Benes, *Mathematical theory of connecting networks and telephone traffic* (Academic Press, New York, 1965).
6. H. S. Hinton, J. R. Erickson, T. J. Cloonan, F. A. P. Tooley, F. B. McCormick, and A. L. Lentine, *An Introduction to Photonic Switching Fabrics* (Plenum Press, 1993), Chap. 3.
7. S. Yao, B. Mukherjee, and S. Dixit, “Advances in photonic packet switching: an overview,” IEEE Commun. Mag. **38**(2), 84–94 (2000).
8. T. Zhang, K. Lu, and J. P. Jue, “Shared fiber delay line buffers in asynchronous optical packet switches,” IEEE J. Sel. Areas Comm. **24**(4), 118–127 (2006).
9. J. H. Kim, J. Y. Choi, M. H. Kang, and J.-K. K. Rhee, “Design of novel passive optical switching system using shared wavelength conversion with electrical buffer,” IEICE Electron. Express **3**(24), 546–551 (2006).
10. K. Padmanabhan and A. Netravali, “Dilated networks for photonic switching,” IEEE Trans. Commun. **35**(12), 1357–1365 (1987).
11. W. Kabacinski, “Modified dilated Benes networks for photonic switching,” IEEE Trans. Commun. **47**(8), 1253–1259 (1999).
12. D. C. Opferman and N. T. Tsao-Wu, “On a class of rearrangeable switching networks part I: Control algorithm,” Bell Syst. Tech. J. **50**, 1579–1600 (1971).
13. C. K. Lee and J.-K. K. Rhee, “Analysis of PLR for shared switch fabrics,” ETRI J. **33**, 136–139 (2011).

1. Introduction

An everlasting growth trend of Internet traffic intrigues a new research question if the network service industry can sustain increasing network energy consumption with the current networking technology. In the optical networking researches, energy efficiency in transporting information becomes one of the major technology concerns. Architecture optimization in designing large-scale optical switches and routers can improve energy efficiency by having

#139661 - $15.00 USD Received 16 Dec 2010; revised 11 Apr 2011; accepted 12 Apr 2011; published 19 Apr 2011

more network traffic processed by energy-efficient functions [1]. For example, cross-layer design optimization between low energy-consuming optical circuit switching and high energy-consuming electronic packet switching can be an interesting problem [2]. Even within the optical technology domain, balancing low energy-consuming passive medium devices and high energy-consuming active medium devices brings another dimension of an energy optimization problem. Passive medium photonic switches such as micro-mechanical, electro-optic, and thermo-optic switches can operate microsecond and millisecond switching, which can provide optical flow switching [3] or packet burst switching with very low energy consumption. Especially, an ideal electro-optic switch device requires only pico-joules of energy per each reconfiguration of the switch state [1,4]. However, the switch scalability of a switching network consisting of typical 2×2 electro-optic switch devices is limited by the architecture complexity. A common network switch design includes add-drop ports for local connections. This paper proposes an efficient design for add-drop ports in a Benes network, which we refer to as add-drop Benes network (ADBN). An ADBN can be the key enabler for energy efficient optical switch node since it provides add-drop ports midst input and output ports with a very small cost. The design requires a very simple modification in a conventional Benes network configuration. We also propose its configuration algorithm for an optical packet switch node. Because proposed architecture operates in the time-slotted manner, the connection capability of Benes network is *nonblocking*, achieving the same capability as Clos network. Therefore, we limit the application of our idea to Benes network because it is more scalable. This work can be easily extended to the similar types of optical switch fabrics.

2. Single ADBN

The Benes network has been studied for decades as a typical example of a *rearrangeably nonblocking network* [5,6]. As a multistage interconnection network, a Benes network can scale with additions of 2×2 switch elements. An $N\times N$ Benes network is constructed by $2\log_2 N$ - 1 stages and $N/2$ switch elements per each stage, so a total of $N(\log_2 N - 1/2)$ switch elements are required. Under the recursive configuration, an $N\times N$ Benes network consists of two $N/2\times N/2$ Benes networks and two outer stages. The Benes permutation network is broadly adopted for a photonic space division switching.

2.1 Architecture

A photonic switching fabric constructed as a Benes network can be utilized in a switching node of optical networks. In the optical packet switching networks, packet contentions occur when packets at different input ports have the same destination output ports in a timeslot. Buffering is required to resolve packet contentions. In order for buffering in a shared buffer, additional add-drop ports are required in a switch fabric. One simple method is to increase the number of ports to $2N\times 2N$ for a degree N node, where N ports are used for link-to-link switching and the rest of N ports are used for add-drop for buffering [7–9]. However, a $2N\times 2N$ Benes network for an add-drop-capable degree-N switch node becomes too costly a solution with poor photonic system performances.

In order to replace the $2N\times 2N$ Benes network, Fig. 1 presents a novel architecture concept of an $N\times N$ ADBN as an example of an 8×8 case, which is an improvement from dilated Benes network [10,11] in terms of 2×2 elements counts. The add-drop function is implemented with the middle stage 2×2 switch elements of a Benes network in a dilated way to form a 4×4 switch fabric and the in-excess ports are used for add-drop multiplex. We define a set of dilated four switches as a *mid-stage*, i.e., there are 4 *mid-stages* newly formed in an 8×8 ADBN and we label them to *mid-stage a*, *b*, *c*, and *d* from the top. There are four types of packet paths in an ADBN, which are input-to-output port (*I-O*), input-to-drop port (*I-D*), add-to-output port (*A-O*), and add-to-drop port (*A-D*). An $N\times N$ ADBN provides add-drop-capable degree-N capacity in total, equivalent to a $2N\times 2N$ Benes network. However, the element savings in an ADBN comes at the price of a limited connection capability.

#139661 - $15.00 USD Received 16 Dec 2010; revised 11 Apr 2011; accepted 12 Apr 2011; published 19 Apr 2011

Nonetheless, the *I-O* path in the ADBN can still achieve the *rearrangeably nonblocking* as a conventional Benes network. However, the connection capability of the *I-D* and *A-O* paths is limited to a *full connection*, i.e., there exist a unique path for each permutation between an *I-D* packet or *A-O* packet. The *A-D* path connection capability is further limited to a *partial connection*, i.e., there exists a unique path for some permutation between an *A-D* packet. The add- and drop-ports can be connected only when both ports belong to the same *mid-stage*.

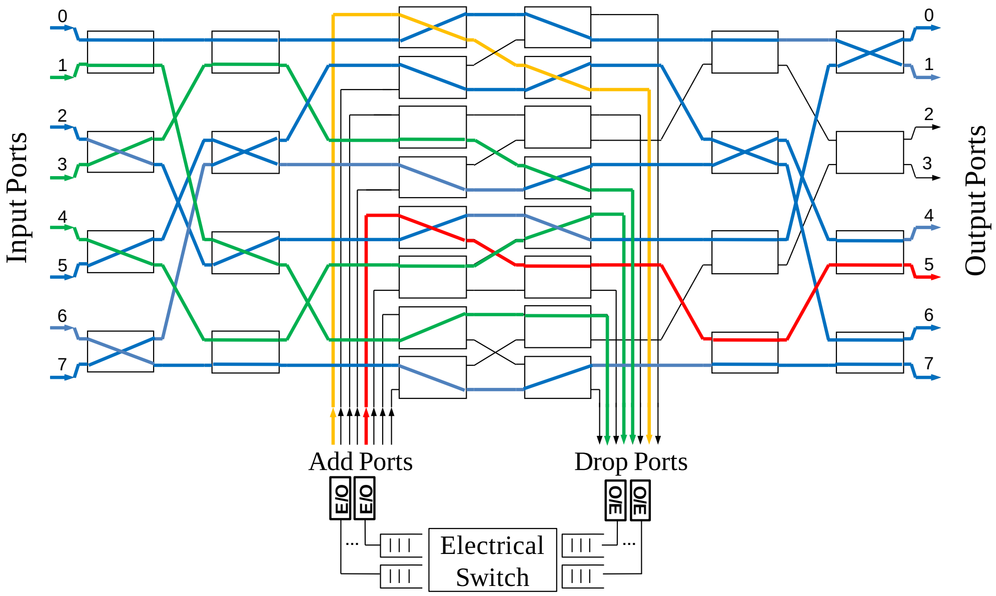

Fig. 1. Architecture of an 8×8 ADBN. Add-drop ports are realized at the middle stage of a conventional Benes network for contention resolution. Colored thick lines indicate the traces of packet schedulings for the example in section 2.2. Here, a limited number of buffers are installed at selected add and drop ports, with a pre-designated buffer installation sequence according to a reconfiguration algorithm.

2.2 Add-drop looping algorithm

Generally a *rearrangeably nonblocking network* requires an appropriate configuration algorithm to provide *nonblocking* switching network configuration in a time slotted switching system. Fortunately, several algorithms have been proposed for a favorable operation of the Benes network. Among them, an algorithm referred to as the looping algorithm is widely adopted because of its simplicity [12]. When any two packets share either input or output stage 2×2 switch, we define a chain as the combination of the two paths of the two packets. In a conventional Benes network, these chains always form a closed loop or loops. The looping algorithm schedules an $N\times N$ Benes network by finding every pair of paths that form a chain. Then each path of a chain is assigned to a different inner sub Benes network in the size of $N/2\times N/2$, by alternating scheduling between two inner subnetworks. The inner subnetworks repeat the same process, recursively [6].

We propose a simple algorithm which is referred to as add-drop looping algorithm (ADLA) for ADBN configuration. The ADLA consists of 5 steps for different path types in every rearrangement timeslot. We introduce the proposed algorithm step by step with the following permutation example for an 8×8 ADBN:

$$P=\begin{pmatrix}0&1&2&3&4&5&6&7\\1&1&0&1&7&4&7&6\end{pmatrix}\qquad\text{and}\qquad P^{add}=\begin{pmatrix}0&5\end{pmatrix}.$$

The permutation P indicates pairs of input port to output port in a given timeslot, where input ports are specified at the upper row and output ports are specified at the bottom. Similarly, P^{add} indicates the destination output ports of add packets which are stored at buffer. We denote $p(m, n)$ as a path from input port m to output port n.

Step1. Classification: The first step of the ADLA classifies packets based on the loop and path types. A classification table is created the order of *I-O* packets, *I-D* packets, and *A-O* packets. After that, it is updated by the each loop. In the current example, there are contentions between packets from input port 0, 1, and 3 since they have the same destination output port. Only one can be scheduled to the output port through an *I-O* path. The others are considered as contended packets and should be dropped through *I-D* paths. Similarly, there is contention between packets from input ports 4 and 6. *A-O* packets can be added only if destination output ports of *A-O* packets are not occupied by the *I-O* packets. Since output port 0 is occupied by an *I-O* packet, the *A-O* packet for destination output port 0 cannot be added in this timeslot. Based on the path types, classification table is created as

$$P=\begin{pmatrix} \overbrace{\begin{matrix}0&2&5&6&7\end{matrix}}^{I-O} & \overbrace{\begin{matrix}1&3&4\end{matrix}}^{I-D} & \overset{A-O}{add}\\ \begin{matrix}1&0&4&7&6\end{matrix} & \begin{matrix}drop&drop&drop\end{matrix} & 5 \end{pmatrix}.$$

In the above classification table, path $p(1, drop)$ can form a chain with path $p(0, 1)$. The path $p(0, 1)$ can form another chain with path $p(2, 0)$. Searching for all connected chains in this example, we can find the following permutation with loop representations. The classification table is updated as

$$P=\left(\overbrace{\begin{bmatrix}0&2&1&3\\1&0&drop&drop\end{bmatrix}}^{loop\ 1}\ \overbrace{\begin{bmatrix}5&4&add\\4&drop&5\end{bmatrix}}^{loop\ 2}\ \overbrace{\begin{bmatrix}6&7\\7&6\end{bmatrix}}^{loop\ 3}\right),$$

which is categorized based on loops. In our ADLA, loops can be either open or closed because of add and drop ports.

Step2. Scheduling for I-O packets: In the ADLA, *I-O* packets are scheduled by the looping algorithm. The connection capability of a *mid-stage* in an ADBN is the same as that of a 2×2 switch element when ADBN switches the packet for input-to-output forwarding. The ADLA provides the same routability for *I-O* packets in ADBN as the looping algorithm used for Benes network [12]. After scheduling the *I-O* packets, the number of occupied paths in *mid-stages* of an $N\times N$ ADBN is the same as the number of the scheduled *I-O* packets. In general, a looping algorithm selects inner subnetworks recursively, alternating upper and lower subnetworks for path assignments. As an example for an 8×8 ADBN, one can select *mid-stages* assignment order of $\{a, c, b, d\}$. The looping algorithm is initiated from an *I-O* path which forms a chain with an *I-D* path. When there is no *I-D* path in a loop, an arbitrary *I-O* path can be selected to begin with. Accordingly, the looping algorithm starts from $p(0, 1)$ in loop 1, as the following classification table of this example, showing *mid-stage* assignments:

$$P=\left(\overbrace{\begin{bmatrix}0&2&1&3\\1&0&drop&drop\\\underline{a}&\underline{c}&&\end{bmatrix}}^{loop\ 1}\ \overbrace{\begin{bmatrix}5&4&add\\4&drop&5\\\underline{b}&&\end{bmatrix}}^{loop\ 2}\ \overbrace{\begin{bmatrix}6&7\\7&6\\\underline{d}&\underline{a}\end{bmatrix}}^{loop\ 3}\right).$$

The bottom line of the classification table indicates the occupied *mid-stages* by the given input to output port pairs. Blue lines in Fig. 1 show the traces of *I-O* packet schedulings.

Step3. Scheduling for I-D packets: After step 2, the non-occupied *mid-stages* are assigned for *I-D* paths by the reverse sequence of step 2. *I-D* packets can be dropped via arbitrarily selected drop ports in *mid-stages*. Therefore in this example, the *I-D* packets are scheduled to pass alternating lower and upper inner subnetworks. With this reverse sequence, the *I-D* packets are scheduled preferably to pass *mid-stages* that have a higher degree of freedom in configuration. Since the connection capability of *I-D* paths satisfies the *full connection*, there exists one and only one non-occupied path between an input port to a *mid-stage*.

When the number of buffers is less than the number of drop ports, so the buffers are shared among ports in a sense of partially shared buffering, we give drop priority to the drop ports that have installed buffers. Hence the buffers should be installed in the order of *mid-stages* assignment sequence of this step. In the current example, there are three *I-D* packets. The *I-D* paths $p(1, drop)$ and $p(3, drop)$ in loop 1 are scheduled to pass *mid-stage* d and b, respectively. The *I-D* path $p(4, drop)$ is scheduled to pass *mid-stage* c. The *mid-stages* in an ADBN can forward the *I-D* packets to available drop ports attached to buffers when drop ports are not used. The classification table is updated as

$$P = \left(\overbrace{\begin{bmatrix} 0 & 2 & 1 & 3 \\ 1 & 0 & drop & drop \\ a & c & \underline{d} & \underline{b} \end{bmatrix}}^{loop\ 1} \; \overbrace{\begin{bmatrix} 5 & 4 & add \\ 4 & drop & 5 \\ b & \underline{c} & \end{bmatrix}}^{loop\ 2} \; \overbrace{\begin{bmatrix} 6 & 7 \\ 7 & 6 \\ d & a \end{bmatrix}}^{loop\ 3} \right).$$

Green lines in Fig. 1 show the traces of *I-D* packet schedulings.

Step4. Scheduling for A-D packets: ADBN can schedule *A-D* packets within *mid-stages*. *A-D* packets occur in a shared ADBN design which will be discussed in Section 3. Because an *A-D* packet is directly added through a *mid-stage*, the drop port is predetermined between two possible drop ports of the *mid-stage*. If there is an arrival of an *A-D* packet, the ADBN can forward it to an available drop port. The orange line in Fig. 1 shows the trace of an *A-D* packet scheduling.

Step5. Scheduling for A-O packets: *A-O* packets can be added through any add port of a *mid-stage*. When the output port for a particular *A-O* packet is not occupied by an *I-O* packet, there exists one and only one available path to reach the destined output port from the *mid-stage* because an *A-O* path achieves *full connection*. Output port checking and scheduling for an *A-O* packet should be conducted by the scheduler of a buffer. In the current example, path $p(add, 5)$ is scheduled with *mid-stage* c. Finally, the classification table for the current example is completed as

$$P = \left(\overbrace{\begin{bmatrix} 0 & 2 & 1 & 3 \\ 1 & 0 & drop & drop \\ a & c & d & b \end{bmatrix}}^{loop\ 1} \; \overbrace{\begin{bmatrix} 5 & 4 & add \\ 4 & drop & 5 \\ b & c & \underline{c} \end{bmatrix}}^{loop\ 2} \; \overbrace{\begin{bmatrix} 6 & 7 \\ 7 & 6 \\ d & a \end{bmatrix}}^{loop\ 3} \right).$$

The red line in Fig. 1 shows the trace of an *A-O* packet scheduling.

3. Shared ADBN

3.1 Architecture

A node architecture model of shared modular switch fabrics and its gain from the points of views of system cost, power consumption [1], and packet loss rate (PLR) [13] are presented. Figure 2 illustrates our shared ADBN node model. This design provides wavelength-layered switching with shared electrical buffering among different wavelength layers. The sharing of buffers between ADBN modules in different wavelength layers can improve contention resolution dramatically due to the better buffer utilization and less system cost. Each wavelength layer uses the contention resolution scheme of the single ADBN, where a part of the add- and drop-ports are used for the buffer interfaces, referred to as *buffer-add ports* and *buffer-drop ports*, respectively. In a shared ADBN, the rest of the add and drop ports are used for the connection from and to ADBNs of neighbor wavelength layers, referred to as *neighbor-add ports* and *neighbor-drop ports*, respectively. In this way, when all the *buffer-drop ports* of an ADBN are busy, contended packets can reach the buffers of neighbor-wavelength ADBNs via *neighbor add/drop ports*.

In order to send a packet that came through *neighbor-add port* to its destined output fiber, the wavelength should be converted from that of the neighbor to that of the current ADBN. This wavelength conversion can be achieved by forwarding the packet to one of the *buffer-drop ports*. Therefore, a packet from a neighbor is always forwarded via *A-D* path. Then the packet is converted to electrical data, stored in a buffer, and scheduled for the destined fiber output port via the transmitter of the wavelength. The 2×2 switch and buffer input interface in an ADBN are broadband devices to handle packets in various wavelengths.

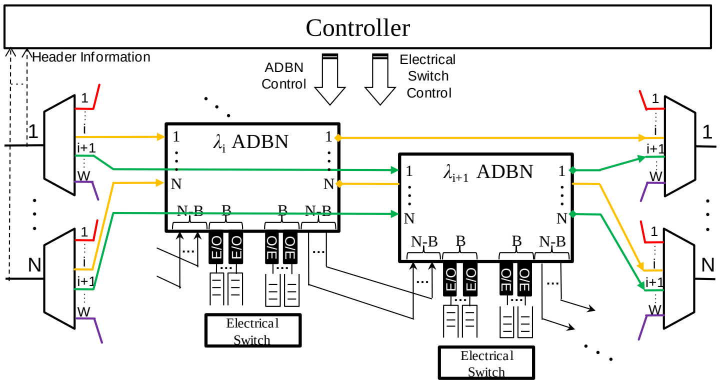

Fig. 2. Architecture schematic of the shared $N \times N$ ADBN in *W* wavelengths channel system. The controller checks the header information of ingress packets. Based on ADLA, the controller determines the configurations of each ADBN and electrical switch.

In Fig. 2, *W* channels from *N* fiber links are switched by ADBNs of the corresponding wavelength layers. A contended packet at λ_i, i.e., in the *i*-th wavelength-layered ADBN is dropped to a *buffer-drop port* if there exist available drop ports for buffer in this ADBN. If there are no available *buffer*-drop ports in this ADBN, the *I-D* packet is dropped to a *neighbor-drop port* toward the (*i* + 1)-th wavelength-layer ADBN. In the (*i* + 1)-th wavelength-layer ADBN, this packet is either sent to a *buffer-drop port* or to a *neighbor-drop port* for the (*i* + 2)-th ADBN, depending on the availability of *buffer-drop ports*. The packet being forwarded from the previous neighbor to the next through a *neighbor-drop port* is referred to as an *A-D* packet. This forwarding behavior can be repeated at the second and later neighbors. Hence, a contended packet can be passed up to *W*-1 times to successive neighbors or until it finds an available *buffer-drop port*; if it fails, the packet is discarded. As the contention scheduling policy gives higher priority to the *buffer-drop* for contended packets, we can reduce the *neighbor-drop* packets for better system performance. *A mid-stage* can forward at most two packets to the drop ports in a timeslot. For when a *mid-stage* receives more than two *I-D* and *A-D* packets, the rest will be lost. This case is referred to as a *mid-stage loss*.

3.2 Performance evaluation

As a photonic switching fabric, the performance of an ADBN can be evaluated by the measures of signal to interference-noise ratio (SINR), insertion loss, and elements counts. The SINR can be enhanced by several architectures of dilated Benes networks [10,11]. This paper follows the performance measure used in [11]. As a good approximation, only the first- and second-order crosstalks are considered since higher-order crosstalks are negligible. Table 1 summarizes the SINR, insertion loss, and required number of 2×2 switches for various types of add-drop-capable degree-*N* Benes networks. Here, variables *X* and *L* measured in the decibel (dB) scale represent for the extinction ratio and the insertion loss of a 2×2 switch, respectively. We assume all 2×2 switches are integrated as a single chip on a planar waveguide substrate. The coupling loss of the waveguide-fiber interface is denoted as *C* in the dB scale.

Table 1. SINR, Insertion Loss, and the Number of Required 2×2 Switch Elements for Various Types of Degree-*N* Add-Drop-Capable Benes Networks $(k=\log_2 M)$

Networks	SINR [dB]	Insertion loss [dB]	Number of switch elements
Benes	$\lvert X\rvert - 10\log(2k-1)$	$(2k-1)L + 2C$	$M\,(2k-1)/2$
Dilated Benes (DB)	$2\lvert X\rvert - 10\log k(2k-1)$	$2kL + 2C$	$2M\,k$
Modified DB	$2\lvert X\rvert - 10\log k(k-1)$	$(2k+1)L + 2C$	$2M\,(k+1)$
ADBN (*I-O*)	$\lvert X\rvert - 10\log 2k$	$2kL + 2C$	$M\,(k+1)$
ADBN (*I-D*, *A-O*)	$\lvert X\rvert - 10\log(k+1)$	$(k+1)L + 2C$	
ADBN (*A-D*)	$\lvert X\rvert - 10\log 2$	$2L + 2C$	

Figure 3 presents comparison of each performance metric as a function of *N*. In order to provide ports for *N* network links and *N* add-drop ports, Benes, dilated Benes, and modified Benes networks must have Benes network equivalent degree of *M* = 2*N*, while an ADBN requires only *M* = *N*. In an ADBN, the *I-D* and *A-O* paths show the same SINR and insertion loss since both paths pass the same number of stages. Because *A-D* path always pass 2 stages, the SINR and insertion loss of *A-D* path have constant values.

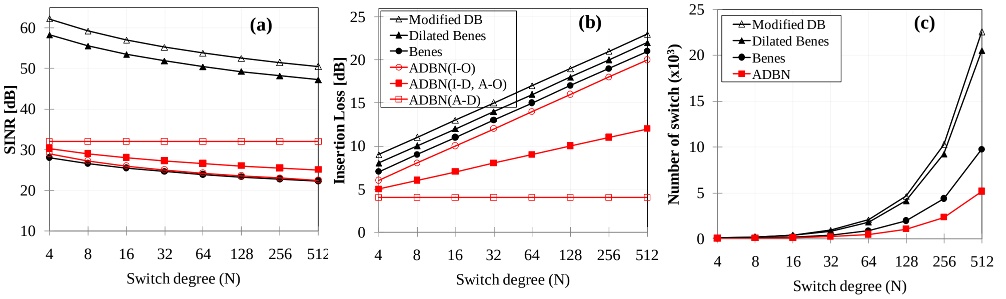

Fig. 3. Comparisons of SINR (a) and insertion loss (b) among ADBN, Benes network, and Dilated Benes networks are plotted as a function of the port counts *N*. As an example, we use the insertion loss *L* of 1dB, coupling loss *C* of 1dB, and extinction ratio *X* of 35 dB. The required number of 2×2 switch elements is shown in (c).

As expected, dilated Benes networks show better SINR performances than that of the ADBN at the price of a large number of additional switch elements leading to more cost and energy consumptions. Comparing with the Benes network, an ADBN shows better SINRs overall in all switch degrees. It is remarkable that the ADBN achieves the least insertion loss by the element savings. Ideally, contended packets can be transferred to neighbor ADBNs successively, up to *W*-1 times in the shared ADBN in *W* wavelength channel system. However, considering the O/E (Optical to Electrical converter) receiver sensitivity at the buffer interface, the number of transfers is limited due to the optical power loss in *A-D* paths. Accordingly, we limit the neighbor transfer up to *k* times, meaning that *A-D* packets transferred more than *k* times are automatically discarded. We refer to *k* as the neighbor transfer cut-off.

As described in [13], shared switch fabric enhances the PLR dramatically. However, the previous analytic model has assumed every switch fabric module as *strictly nonblocking*

#139661 - $15.00 USD Received 16 Dec 2010; revised 11 Apr 2011; accepted 12 Apr 2011; published 19 Apr 2011

network. There are two types of packet loss events in the previous model. The first type of packet loss happens within a switch fabric when the sum of *I-D* and *A-D* packets is larger than the total number of drop ports of the switch fabric. The second type of packet loss happens when the total of contended packets in all switches are larger than the total of installed buffers. However, in the shared ADBN case, the major events of packet loss happen due to *mid-stage loss* and neighbor transfer cut-off. We define *cut-off loss* for packet loss event caused by neighbor transfer cut-off. For the effective PLR calculation, we evaluate the packet loss at an *i*-th $N \times N$ ADBN as follow,

$$PLR_{N,B,k}^{shared\ ADBN} = \frac{\lambda_{loss}^{i}(k)}{N\rho} \qquad for\ k < W, \tag{1}$$

where $\lambda_{loss}^{i}(k)$ is the average total number of *mid-stage loss* and *cut-off loss* at the *i*-th ADBN. Here, *k* is the neighbor transfer cut-off, and *W* is the total number of layered .. ADBNs. Each ADBN has *B buffer-drop ports*. An offered load at a node is denoted as ρ. The $\lambda_{loss}^{i}(k)$ is calculated as nested summations of

$$\lambda_{loss}^{i}(k) = \sum_{x_i=0}^{N-1} P(X = x_i) \sum_{x_{i-1}=0}^{N-1} P(X = x_{i-1}) \cdots \sum_{x_{i-k}=0}^{N-1} P(X = x_{i-k}) \psi_{loss}^{(i,k)}(x_i, \cdots, x_{i-k}), \tag{2}$$

where *P(X)* indicates the probability of *X* number of packet contentions [13], and x_j is the number of contended packets at the *j*-th ADBN. Sub-index (*i* – *k*) is wrapped around by modulo of *N*. $\psi_{loss}^{(i,k)}(x_i, \cdots, x_{i-k})$ represents the total number of *mid-stage loss* and *cut-off loss*.

We define the number of contended packets at the *i*-th ADBN as $\chi_{I-D}^{i} = x_i$. Similarly, we define $\chi_{A-D}^{(i,k)}$ to the number of *A-D* packets at the *i*-th ADBN with the neighbor transfer cut-off *k*. Then, the (*i*-1)-th ADBN should forward $\chi_{I-D}^{i-1} + \chi_{A-D}^{(i-1,k)}$ number of packets to its drop ports. These *I-D* and *A-D* packets can be lost due to the *mid-stage loss* or *cut-off loss* at the (*i*-1)-th ADBN. We give priority for drop toward buffer, so the number of *A-D* packets at the *i*-th ADBN can be counted by following recursive form:

$$\chi_{A-D}^{(i,k)} = [\chi_{I-D}^{i-1} + \chi_{A-D}^{(i-1,k)} - \psi_{loss}^{(i-1,k)}(x_{i-1}, \cdots, x_{i-1-k}) - B]^{+}, \tag{3}$$

where $[A]^{+} \equiv \max(A, 0)$ for real number *A*. In order to evaluate (3), an initial condition is required. In our evaluation, we assumed $\chi_{A-D}^{(i-k,k)} = 0$ for evaluation of $\chi_{A-D}^{(i,k)}$ based on empirical observations. Within the parameter space considered in our study, we find this approximation is reasonably acceptable.

$\psi_{loss}^{(i,k)}(x_i, \cdots, x_{i-k})$ can be calculated as sum of numbers of *mid-stage loss* and *cut-off loss*. As described, *mid-stage loss* occurs when more than two *I-D* and *A-D* packets are forwarded to pass a specific *mid-stage*. There are *N*/2 *mid-stages* in an $N \times N$ ADBN and the ADLA schedules the *I-D* packets to pass *mid-stages* in step 3. When the number of *I-D* packets is equal or smaller than the number of *mid-stages* in the ADBN, the *I-D* packets are schedule to pass χ_{I-D} number of *mid-stages* at once. When the number of *I-D* packets is larger than *N*/2, the *I-D* packets are scheduled to pass $\chi_{I-D} - N/2$ number of *mid-stages* at twice and $N - \chi_{I-D}$ number of *mid-stages* at once. The *A-D* packets also follow the aforementioned pattern. Because there is no scheduler inside the *A-D* paths, the sequence of *mid-stage* assignment of *A-D* packets is determined by the same to that of *I-D* packets. Therefore, the number of *mid-stage loss* $\psi_{mid\_stage}^{(i,k)}$ can be expressed as

$$\psi^{(i,k)}_{mid\_stage} = \begin{cases} 0 & \chi^{i}_{I-D} \le N/2 \text{ and } \chi^{(i,k)}_{A-D} \le N/2 \\ \chi^{i}_{I-D} - N/2 + \chi^{(i,k)}_{A-D} - N/2 & \chi^{i}_{I-D} > N/2 \text{ and } \chi^{(i,k)}_{A-D} > N/2 \\ \min[\chi^{i}_{I-D} \bmod (N/2),\, \chi^{(i,k)}_{A-D} \bmod (N/2)] & \text{otherwise} \end{cases} \tag{4}$$

We denote $\psi^{(i,k)}_{cut-off}$ for the number of *cut-off loss* at the *i*-th ADBN with neighbor transfer cut-off *k*. In order to calculate $\psi^{(i,k)}_{cut-off}$, we tag the ADBN origin to *A-D* packets to track the *A-D* packets for cut-off decision. In this way, one can count $\psi^{(i,k)}_{cut-off}$ by investigating all *A-D* packets. After counting $\psi^{(i,k)}_{cut-off}$ and $\psi^{(i,k)}_{mid\_stage}$, we can calculate $\psi^{(i,k)}_{loss} = \psi^{(i,k)}_{mid\_stage} + \psi^{(i,k)}_{cut-off}$.

For a practical application, we consider a shared 8×8 ADBN in a *W*-wavelength-channel system. In order to count the packet loss events effectively, we generate every possible packet contention one by one in each ADBN and exhaustively counts packet loss events. Table 2 shows the minimum required *buffer-drop ports* counts to achieve less than a PLR of 10^{-6} in a shared 8×8 ADBN node at various offered loads. For this example, we consider three different policies where the neighbor transfer cut-off is set at the first, second, or third neighbor (*k* = 1, 2, or 3) in a system with *W*>3.

Table 2. Minimum Number of Required Buffer-Drop Ports to Achieve 10^{-6} PLR with Various Offered Load for Shared 8×8 ADBN*

Cut-off of neighbor transfer	Offered load (ρ)								
	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
First neighbor	2	3	3	4	4	4	5	5	5
Second neighbor	2	2	3	3	4	4	4	5	5
Third neighbor	1	2	2	3	3	4	4	4	5

*The first, second, and third neighbor cut-off transfer policies are investigated.

When we limit neighbor transfer to the first, second, and third, Table 2 shows numbers of required *buffer-drop ports* equivalent to those of 2, 3, and 4-shared switch fabrics of the analytic model in [13], respectively. As observed by the comparisons with the previous analytic model, it is clear that the connection capability limitations of the ADBN have negligible effect on buffer counts by proposed algorithm and architectural rules.

4. Conclusions

This paper proposes a photonic switch design of a novel ADBN for energy and cost efficient optical networks. A shared ADBN node architecture model that can share buffers with each other is investigated. In a practical application, a shared ADBN node can be used for a tera-bit-scale WDM optical packet transport network with reasonable crosstalk and optical loss penalties, for example, a WDM network node consisting of 8 fiber links, 80 wavelengths per fiber links, and 40Gbps per wavelengths. This photonic switching ADBN extends the Benes network with add-drop ports with up to 50% switch element savings compared with that of an equivalent Benes network. We proposed an appropriate algorithm and the corresponding architecture rule to provide the adequate packet routability and buffer sharing with other ADBNs in the node. Considering the comparisons of crosstalk and insertion loss penalty with other Benes network models, the proposed ADBN designs manifest outstanding scalability.

Acknowledgments

This research was supported by the MKE (The Ministry of Knowledge Economy), Korea, under the ITRC (Information Technology Research Center) support program supervised by the NIPA (National IT Industry Promotion Agency) (NIPA-2010-(C1090-1011-0004)).