HAL
open science

An Explicit Control Algorithm for Optical FIFO Queues

Anne Bouillard, Cheng-Shang Chang

To cite this version:

Anne Bouillard, Cheng-Shang Chang. An Explicit Control Algorithm for Optical FIFO Queues. [Research Report] RR-6097, 2007, pp.11. inria-00123901v1

HAL Id: inria-00123901

https://inria.hal.science/inria-00123901v1

Submitted on 11 Jan 2007 (v1), last revised 12 Jan 2007 (v2)

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

An Explicit Control Algorithm for Optical FIFO Queues

Anne Bouillard and Cheng-Shang Chang

N° ????

Janvier 2007

Thème COM

Rapport de recherche

ISSN 0249-6399 ISRN INRIA/RR--????--FR+ENG

An Explicit Control Algorithm for Optical FIFO Queues

Anne Bouillard* and Cheng-Shang Chang†

Thème COM — Systèmes communicants
Projet Distribcom

Rapport de recherche n° ???? — Janvier 2007 — 11 pages

Abstract: With the recent advances in optical technologies, it has become a challenge to build optical queues with minimal complexity. In [2], it was shown that an optical FIFO queue can be constructed recursively by a concatenation of scaled optical memory cells, which in turn are made by 2×2 switches and fiber delay lines. However, as the construction is recursive, there is no explicit control algorithm for the 2×2 switches in [2]. The main contribution of this paper is to provide an explicit control algorithm for the 2×2 switches in that construction. We show that our algorithm has $O((\log B)^2)$ space complexity and time complexity for an optical FIFO queue with buffer B.

Key-words: Optical networks, algorithm

* IRISA/ENS Cachan (Bretagne), Campus de Ker Lann, 35170 BRUZ, France. Email: Anne.Bouillard@bretagne.ens-cachan.fr.
† Institute of Communications Engineering, National Tsing-Hua University, Hsinchu 300, Taiwan R.O.C. Email: cschang@ee.nthu.edu.tw. This research was supported in part by the National Science Council, Taiwan, R.O.C., under Contract NSC-94-2213-E-007-046, and the Program for Promoting Academic Excellence of Universities NSC 94-2752-E-007-002-PAE.

Unité de recherche INRIA Rennes
IRISA, Campus universitaire de Beaulieu, 35042 Rennes Cedex (France)
Téléphone : +33 2 99 84 71 00 — Télécopie : +33 2 99 84 71 71

Un algorithme de contrôle explicite pour les files d'attentes FIFO optiques

Résumé : Avec les avancées récentes dans le domaine des réseaux optiques, chercher à construire des files d'attente optiques de complexité minimale s'annonce prometteur. Dans [2], il a été montré qu'une file d'attente FIFO optique peut être construite de manière récursive par la concaténation de cellules optiques, qui sont constituées de routeurs optiques 2×2 et de fibres optiques. Mais, comme la construction donnée est récursive, on ne peut pas en déduire directement un algorithme de contrôle des cellules optiques. La principale contribution de ce rapport est de fournir un algorithme de contrôle explicite des cellules. Nous montrons que cet algorithme a une complexité spatiale et temporelle en $O((\log B)^2)$ où B est la capacité de la file d'attente.

Mots-clés : Réseaux tout-optique, algorithmes

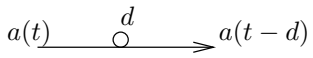

Figure 1: A delay line with delay d.

1 Introduction

An important issue in communication systems is to resolve the conflicts for packets competing for the same resource. Traditionally, this is made by switching and queuing packets in some network elements, switches and buffers. Those elements make use of electronic memories. However, the recent development of optical networks leads to a very high transmission rate based on photons, and the transmission rate of electronic memories simply cannot keep up with that of optical networks.

Two solutions have been proposed to overcome this problem. The first one is to use parallel switches so that the transmission rate is higher (see e.g., [8] and references therein), but this becomes very costly, and one needs to make optical-electronic and electronic-optical conversions. The second solution is to design optical network elements and resolve the conflicts on optical packets directly. The main difficulty of this approach is that optical packets can not be stored as easily as electronic packets because of their photonic nature (packets cannot be "stopped"). They always move, and one has to make them circulate in a network element in such a way that they can exit the system at the requested time.

One of the approaches that has already appeared in [6, 4, 10, 7] is to make use of optical switches and fiber delay lines. The switches enable the control of the packets and the delay lines make the packets circulate in the system so that they are stored and waiting for their departure. The challenge is then to design network elements and find the corresponding control algorithms to efficiently emulate the network elements that already exist for electronic packets. There already exist some designs that can emulate some network elements, such as FIFO multiplexers in [6, 3, 5], FIFO queues in [2], delay lines in [1], and priority queues in [9].

In [2], it was shown that an optical FIFO queue can be constructed recursively by a concatenation of scaled optical memory cells, that in turn are made by 2×2 switches and fiber delay lines. However, as the construction is recursive, there is no explicit control algorithm for the 2×2 switches in [2]. The main contribution of this paper is to provide an explicit control algorithm for the 2×2 switches in that construction. Our main idea is to represent a scaled optical memory cell with scaling factor B by B separate buffers. Each of them is capable of holding exactly one packet. By so doing, we are able to map the optical FIFO queue in [2] to a network of buffers with *periodically* activated parallel paths. By introducing the concept of target buffer, a packet, when activated, is then routed to the buffer that is closest to its target buffer. We show that such an algorithm has $O((\log B)^2)$ space complexity and time complexity for an optical FIFO queue with buffer B.

2 Basic elements and FIFO queues

In this section, we introduce more precisely the elementary components of optical network elements and the construction for the FIFO queue in [2]. Throughout the paper, we assume that packets are of the same size. Moreover, time in all the optical links is slotted and synchronized so that a packet can be transmitted within a time slot. A *state* of a link at each time t ($t = 0, 1, 2, \ldots$) is either 0 or 1. It is 1 if there is a packet in the link at time t and 0 if there is no packet.

Definition 1. *A* delay line *with delay d is a network element that has one input and one output, such that, if at time t the state of the input link is $a(t)$, then the state of the output link is $a(t-d)$. Fig. 1 represents a delay line with delay d.*

Delay lines are very useful to store packets in optical networks: there can be d packets stored in a line with delay d. Each of them remains stored exactly d units of time. The other elementary element of optical networks is the 2×2 *optical crossbar switch.*

Definition 2. *A 2×2 optical crossbar switch is a network element that has two inputs and two outputs. If at time t the input state is $(a_1(t), a_2(t))$, there can be two different output states, $(a_1(t), a_2(t))$ (the switch is in the* bar *state) or $(a_2(t), a_1(t))$ (the switch is in the* cross *state).*

An *optical memory cell* is the combination of those two basic elements, and when the delay line has delay 1, it can be operated as a memory cell. The bar state is used to store the information whereas the cross state

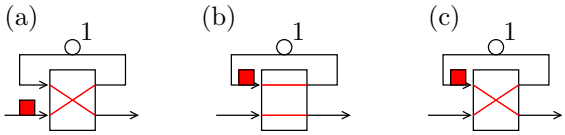

Figure 2: Optical memory cell: (a) writing information (b) circulation information (c) reading information.

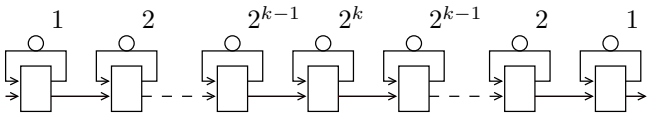

Figure 3: A FIFO queue with buffer $2^{k+1} - 1$.

is used for the writing and reading operations, as shown in Fig. 2. If the delay line of such a combination is $d \in \mathbb{N} \setminus \{0, 1\}$, then it is called a *scaled optical memory cell* with scaling factor d. By abuse of the language, we will still call it an optical memory cell.

Now we introduce the definition of a FIFO queue and its construction in [2].

Definition 3. *A FIFO queue with buffer B is a network element with one input link (for the arrivals), one control input and two output links (one for the departures and one for the losses) of respective states $a(t)$, $c(t)$, $d(t)$ and $\ell(t)$ at time t. Let $q(t)$ be the number of packets in the system at time t. Then, the FIFO queue must satisfy the following four properties.*

(P1) Flow conservation: *arriving packets are either stored in the buffer, or transmitted through the two output links:*

$$q(t) = q(t-1) + a(t) - d(t) - \ell(t).$$

(P2) Non-idling: *if the control input is enabled ($c(t) = 1$), then there is always a departing packet, unless the system is empty and there is no arriving packet:*

$$d(t) = \begin{cases} 1 & \text{if } c(t) = 1 \text{ and } q(t-1) + a(t) > 0 \\ 0 & \text{otherwise.} \end{cases}$$

(P3) Maximum buffer usage: *if the control input is not enabled ($c(t) = 0$), then an arriving packet is lost only when the buffer is full:*

$$\ell(t) = \begin{cases} 1 & \text{if } c(t) = 0,\ q(t-1) = B \text{ and } a(t) = 1 \\ 0 & \text{otherwise.} \end{cases}$$

(P4) FIFO: packets depart in the first in first out (FIFO) order.

It is shown in [2] that a concatenation of $2k+1$ scaled optical memory cells (with scaling factors $1, 2, \ldots, 2^{k-1}, 2^k, 2^{k-1}, \ldots,$ in Figure 3 can be operated as a FIFO queue with buffer $2^{k+1} - 1$. However, there is no explicit control algorithm for the $2k+1$ 2×2 switches in the construction. In the next section, we will provide an explicit control algorithm for the $2k+1$ 2×2 switches.

Note that in Figure 3, we only represent one output link for the departures, as the losses can be determined upon the arrivals of the packets, and it is possible to get rid of the lost packets with a 2×2 optical crossbar switch placed before the construction.

3 An explicit control algorithm for the FIFO queue

In this section, we give an explicit control algorithm for the construction in Figure 3 to operate as a FIFO queue.

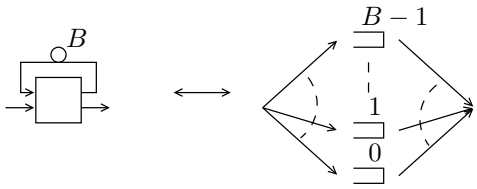

Figure 4: Representation of an optical memory cell with scaling factor B.

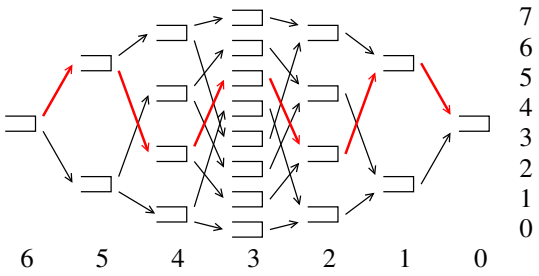

Figure 5: Network representing Fig. 3 with buffer 15. The red line is line 5.

3.1 Active lines and target buffers

We first introduce the concepts of *active lines* and *target buffers*. As a scaled optical memory cell with scaling factor B (see Figure 4) can be used for storing B packets, it can be viewed as a network element that consists of B separate buffers. Each of them is capable of holding exactly one packet. However, these B buffers can only be accessed *periodically* with period B because of the sequential nature of the fiber delay line with delay B. Specifically, if we index these B buffers from 0 to $B-1$, then buffer j is *active* (that is, can be accessed) at time t if and only if $t \mod B = j$.

With such a representation, we can then represent the construction in Fig. 3 by a network of buffers. Specifically, we index the optical memory cells from right to left from 0 to $2k$. Let $\bar{i} = \min(i, 2k-i)$. Since the ith optical memory cell is with scaling factor $2^{\bar{i}}$, it consists of $2^{\bar{i}}$ separate buffers. For these buffers, we denote by $b_i(j)$ the jth buffer of cell i, for $j = 0, 1, \ldots, 2^{\bar{i}} - 1$, and buffer $b_i(j)$ is active at time t if and only if $t \mod 2^{\bar{i}} = j$. Sometimes, by abuse of the notation, we still denote by $b_i(j)$ the buffer $b_i(j \mod 2^{\bar{i}})$ when j is larger than $2^{\bar{i}}$.

Note that a packet stored in a buffer in cell i can be moved forward to another buffer in cell $i-1$ if these two buffers can be active at the same time. For this, we connect these two buffers by a directed link (denoted by $\rightarrow$). Since buffer $b_i(j)$ is active at time t when $t \mod 2^{\bar{i}} = j$, we have the following connections:

(i) For $i < k$ and $j < 2^i$, $b_{i+1}(j) \rightarrow b_i(j)$ and $b_{i+1}(j + 2^i) \rightarrow b_i(j)$.

(ii) For $i > k$ and $j < 2^{2k-i}$, $b_i(j) \rightarrow b_{i-1}(j)$ and $b_i(j) \rightarrow b_{i-1}(j + 2^{\bar{i}})$.

To illustrate these connections, we plot the network of buffers for $k = 3$ in Fig. 5. With this representation, it is easy to see that there are exactly 2^k paths of length $2k$: there is exactly one path from cell $2k$ on the left to a buffer of the central cell k and exactly one path from a buffer in the central cell k to the right cell 0. Then there is a one-to-one correspondence between the 2^k paths of length $2k$ and the 2^k buffers in the central cell. We call those paths the *lines* of the network. Let us number them according to the buffer of the central cell they contain. Specifically, for $j = 0, 1, \ldots, 2^k - 1$, line j consists of the sequence of buffers $b_i(j \mod 2^{\bar{i}})$, $i = 2k, 2k-1, \ldots, 1, 0$. Clearly, if two buffers $b_{i_1}(j_1)$ and $b_{i_2}(j_2)$ are on the same line, say line j, then $b_{i_1}(j_1+1)$ and $b_{i_2}(j_2+1)$ are on line $j+1 \mod 2^k$.

Lemma 1. *At time t, a buffer is active if and only if it belongs to line $t \mod 2^k$.*

Proof. We have to show that all the active buffers are on the same line. Suppose that $t \mod 2^k = j$. Then the central buffer that is active is $b_k(j)$ and the active line at time t is line j. Recall that line j consists of the sequence of buffers $b_i(j \mod 2^{\bar{i}})$, $i = 2k, 2k-1, \ldots, 1, 0$. Also, as $\bar{i} \leq k$, we have

$$\begin{aligned} j \mod 2^{\bar{i}} &= (t \mod 2^k) \mod 2^{\bar{i}} \\ &= t \mod 2^{\bar{i}}. \end{aligned}$$

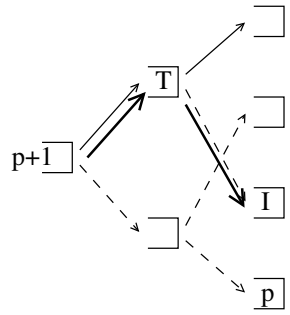

Figure 6: Target buffer.

Thus, all the buffers on line j are active at time t. □

Definition 4. *The* active *line at time t is the line where every buffer on that line is active at time t.*

From Lemma 1, the active line at time t is then the line $t \mod 2^k$, and it consists of the sequence of buffers $b_i(t \mod 2^{\bar{i}})$, $i = 2k, 2k-1, \ldots, 1, 0$. Clearly, every line is active *periodically* with period 2^k. For example, line 5 in Figure 3 consists of the sequence of buffers: $b_6(0)$, $b_5(1)$, $b_4(1)$, $b_3(5)$, $b_2(1)$, $b_1(1)$, and $b_0(0)$. It is active at $t = 5, 13, 21, \ldots$.

We note that at each time slot, the only packets that can be moved are the packets stored in the buffers on the active line. As such, the states of the optical cells (*cross* or *bar*) affect only the packets on the active line at a given time. Suppose that there is a packet in the active buffer in cell i at time t. If cell i is in the bar state, then that packet remains in cell i (in the same buffer). If cell i is in the cross state, then the packet is moved to the next cell j, $j < i$ that is in the cross state. If there is no such cell, then the packet departs from the system.

The next question is then where we move the packets stored in the buffers on the active line. For this, we introduce the concepts of *target buffer* and *target cell*.

For clarity of our purpose, we number the packets according to their arrival order. Suppose that packet p is in buffer $b_i(j)$ at time $t-1$ (the end of the $t-1^{th}$ time slot). The *ideal buffer* for the packet $p+1$ at time t is defined to be $b_i(j+1)$ (or more precisely $b_i(j+1 \mod 2^{\bar{i}})$). This is because if packet p is moved at some time $t_1 \geq t$, then buffer $b_i(j+1)$ will be active at time $t_1 + 1$ and packet $p+1$ (if stored in the ideal buffer) can be moved just after packet p. By so doing, the FIFO order can be preserved. If packet p is not in the system at time $t-1$, then the ideal buffer for packet $p+1$ at time t is defined to be $b_0(0)$.

Intuitively, we should move packet $p+1$ to its ideal buffer when the buffer holding packet $p+1$ is active. However, its ideal buffer may not be on the active line, and we can only move packet $p+1$ to the closest buffer (the buffer that has the longest common sub-path between the active line and the path to its ideal buffer). The closest buffer on the active line to the ideal buffer is then called the *target buffer*. The *target cell* is the cell the target buffer belongs to.

Fig. 6 illustrates the definition of the target buffer. The plain path is the active line. The symbol 'I' represents the ideal buffer, and the symbol 'T' represents the target buffer.

Lemma 2. *Suppose that the ideal buffer of a packet is $b_i(j)$.*

(i) *If $j \mod 2^{\bar{i}} = t \mod 2^{\bar{i}}$, then its target buffer at time t is its ideal buffer $b_i(j)$.*

(ii) *If $j \mod 2^{\bar{i}} \neq t \mod 2^{\bar{i}}$, then its target buffer is $b_{i^*}(j \mod 2^{2k-i^*})$, where i^* is the smallest cell index such that $i^* > \max[i, 2k-i]$ and*

$$j \mod 2^{2k-i^*} = t \mod 2^{2k-i^*}. \tag{1}$$

Proof. Clearly, if the ideal buffer is active at time t, i.e., $j \mod 2^{\bar{i}} = t \mod 2^{\bar{i}}$, then the target buffer is the same as the ideal buffer. Now suppose that the ideal buffer is not active at time t. For $i \geq k$, there is only a unique path from $b_{2k}(0)$ to $b_i(j)$, and this path consists of the sequence of buffers $b_\ell(j \mod 2^{2k-\ell})$, $\ell = 2k, 2k-1, \ldots, i$. Since the active line consists of the sequence of buffers $b_\ell(t \mod 2^{\bar{\ell}})$, $\ell = 2k, 2k-1, \ldots, 1, 0$, the target buffer,

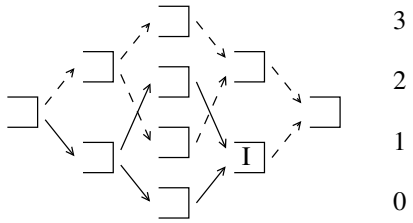

Figure 7: An illustration for the results in Lemma 2.

that is the closest buffer on the active line to the ideal buffer, is buffer $b_{i^*}(j \mod 2^{2k-i^*})$, where i^* is the smallest cell index satisfying (1).

If $i < k$, then $b_{2k-i}(j)$ is on every path from $b_{2k}(0)$ to $b_i(j)$. As $b_i(j)$ is not active at time t, buffer $b_{2k-i}(j)$ is also not active at time t. Thus, the target buffer is on the unique path from $b_{2k}(0)$ to $b_{2k-i}(j)$, and it can be found similarly as for the case $i \geq k$. □

We illustrate the results in Lemma 2 in Fig. 7: (i) the target buffer is the ideal buffer when the active line is 0 or 2, (ii) the target buffer is $b_4(0)$ when the active line is 1 or 3.

For both cases in Lemma 2, we have the following corollary.

Corollary 1. *The target buffer is on every path from $b_{2k}(0)$ to the ideal buffer.*

3.2 Routing algorithm

The main idea of the routing algorithm is then to route a packet to its target buffer each time it is on the active line, with one restriction when the target buffer is occupied. When the target buffer is occupied, the packet is routed to the empty buffer that is closest to the target buffer on the active line.

We first show that our routing algorithm based on such an idea is feasible.

Lemma 3. *For every packet in the system, there is a path from the buffer holding the packet to its ideal buffer.*

Lemma 3 implies that the target buffer of a packet is in the path between the buffer that is holding the packet and the ideal buffer (when the packet is on the active line). Without this, our routing algorithm cannot be feasible as packets can only be moved forward.

Proof. Suppose that packet $p+1$ is in the system and packet p is not. Then the ideal buffer is $b_0(0)$. Since there is a path from any buffer to $b_0(0)$, the result follows trivially in this case.

Now suppose packet p is in $b_{i_1}(j_1)$ when packet $p+1$ arrives. Then the ideal buffer for packet $p+1$ is $b_{i_1}(j_1+1)$. According to our routing algorithm, packet $p+1$ is routed to a buffer, say $b_{i_3}(j_3)$, on the path between $b_{2k}(0)$ and its target buffer, say $b_{i_4}(j_4)$. Now suppose that packet p is moved (maybe several times) to another buffer $b_{i_2}(j_2)$. Then we know (i) the new ideal buffer for packet $p+1$ is then $b_{i_2}(j_2+1)$, and (ii) there is a path from $b_{i_1}(j_1)$ and $b_{i_2}(j_2)$. Without loss of generality, assume line j (for some j) passes through $b_{i_1}(j_1)$ and $b_{i_2}(j_2)$. Then $b_{i_1}(j_1+1)$ and $b_{i_2}(j_2+1)$ are on line $j+1 \mod 2^k$. Thus there is a path from $b_{i_1}(j_1+1)$ to the new ideal buffer $b_{i_2}(j_2+1)$. Since $b_{i_1}(j_1+1)$ is the old ideal buffer of packet $p+1$, it follows from Corollary 1 that there is a path that passes $b_{2k}(0)$, $b_{i_3}(j_3)$, $b_{i_4}(j_4)$, and $b_{i_1}(j_1+1)$. As such, there is a path that passes the buffer holding packet $p+1$ (i.e., $b_{i_3}(j_3)$) and the new ideal buffer $b_{i_2}(j_2+1)$ after packet p is moved. □

By the definition of the target cell, packet $p+1$ can be moved only between its actual cell and the cell occupied by packet p, and an arriving packet can only be routed between cell $2k$ and the cell of the last packet. As such, there is a order for the packets in the buffers.

(P5) Packet order: at any time t, the packets are ordered by the index of the cells, that is, if packet p is in cell i, then packet $p+1$ is in cell $i' \geq i$.

Moreover, observe that the ideal buffer of packet $p+1$ is always next to the buffer of packet p (if packet p is in the system). If packet $p+1$ is moved to the cell that holds packet p, then packet $p+1$ and packet p are stored contiguously. This leads to the following property of our routing algorithm.

(P6) Contiguity: packets in one cell are ordered and contiguous, that is, if the packets in cell i are the packets numbered from p to $p+\ell-1$, $\ell \leq 2^{\bar{i}}$, then there exists j such that for every $r < \ell$, packet $p+r$ is in buffer $b_i(j+r)$ (or more precisely $b_i(j+r \mod 2^{\bar{i}})$).

We note that if packet p is stored in buffer $b_i(j)$ and some packet $p' < p$ is stored in buffer $b_i(j+1)$, then the contiguity property in (P6) implies that packet p' is in fact packet $p - 2^{\bar{i}} + 1$ and cell i is full (i.e., all the buffers $b_i(j)$, $j = 0, 1 \ldots, 2^{\bar{i}}$ are occupied).

In the following lemma, we show that a packet can only be moved to two places: (i) its target buffer and (ii) a predecessor buffer of its ideal buffer.

Lemma 4. *Suppose that packet p is on the active line at time t. Packet p is either stored in its target buffer or in the predecessor buffer of its ideal buffer on the active line at time t. In the second case, the cell that contains its ideal buffer is also its target cell and that cell is full at time t.*

Proof. We have from Lemma 3 that the target buffer of packet p is in the path between the buffer that is holding packet p and its ideal buffer. If its target buffer is the same as the buffer that is holding packet p, then this is the trivial case as packet p remains in that buffer and thus in its target buffer. Excluding this trivial case, let us consider the case that its target buffer is not the same as the buffer that is holding packet p. From (P5), we know that all the packets that are ahead of packet p must be stored in the cells ahead of its ideal buffer. If its target buffer is not same as its ideal buffer, then its target buffer is empty and packet p is moved to its target buffer at time t. Thus, the only reason that packet p is not put in its target buffer is because its target buffer is the same as its ideal buffer and its ideal buffer is occupied. When this happens, packet p is put in the predecessor buffer of its ideal buffer on the active line. Moreover, from the contiguity property in (P6) (and the comment below (P6)), it follows that the cell that contains its ideal buffer is full. □

In view of (P6), one can define *the first packet in a cell* as the packet that has the lowest index among all the packets stored in that cell. From (P5), we know that only the first packet in a cell can be moved when it is on the active line. Moreover, we know from Lemma 4 that the first packet can only be routed to its target cell or the predecessor of its target cell. This leads to the following explicit routing algorithm for a FIFO queue in Algorithm 1.

Algorithm 1: FIFO routing algorithm for each time slot t.

```
begin
    Let i be the first non-empty cell (from the right);
    if the control is enabled, i.e., c = 1 then
        Route the packet in cell i outside of the system;
    else
        Route that packet in cell i to cell 0;
    while i ≤ 2k do
        Let i be the next cell whose first packet is on the active line;
        s ← the target cell for the first packet in cell i;
        if cell s is full then s ← s + 1;
        Route packet in cell i to cell s;
    if there is an arrival, i.e., a = 1 then
        Let s be the target cell of the arriving packet;
        if cell s is full then s ← s + 1;
        Route the arriving packet to s;
end
```

We note that routing a packet from cell i to cell j ($j < i$) in Algorithm 1 can be done by setting both the 2×2 switches of cell i and cell j to the cross state and all the 2×2 switches between cell i and cell j to the bar state. In the case that $i = j$, we can simply set the 2×2 switch of cell i to the bar state so that the packet in cell i remains in the same buffer.

Example 1. *In this example, we illustrate the routing algorithm for a FIFO queue with buffer 7 (i.e., $k = 2$). Consider the arrival process $a(0) = a(1) = a(2) = a(3) = a(5) = a(6) = a(7) = 1$ and $a(4) = a(8) = 0$ and the control process $c(0) = c(1) = c(2) = c(3) = c(6) = c(7) = c(8) = 0$ and $c(4) = c(5) = 1$. The evolution of a*

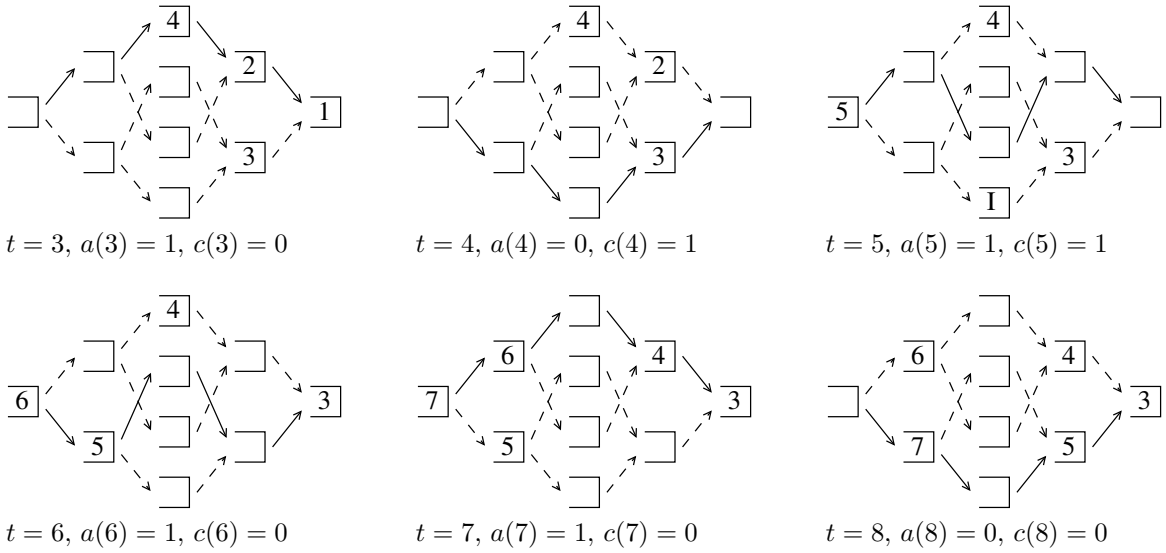

Figure 8: Evolution of a FIFO queue.

FIFO queue with buffer 7 is shown in Fig. 8. The plain lines are the active lines. *As the first packet arrives at time 0, this packet is stored in* $b_0(0)$. *Then the second (resp. third) packet arrives at time 1 (resp. 2) and it is stored in* $b_1(1)$ *(resp.* $b_1(0)$*). The fourth packet arrives at time 3 and it is stored in* $b_2(3)$. *At time 4, the control input is enabled and the first packet leaves the system. The other packets remain in their buffers. At time* 5, *the control input is also enabled and the second packet leaves the system. Packets 3 and 4 remain in their buffers. For packet 5, its ideal buffer is* $b_2(0)$, *marked with symbol 'I'. The target buffer of packet 5 is* $b_4(0)$ *and it is stored there. At time 6, the ideal buffer of packet 5 is still* $b_2(0)$, *but its target buffer is changed to* $b_3(0)$. *Thus, packet 5 is moved to* $b_3(0)$ *at time 6. Packet 6, arriving at time 6, is stored in* $b_4(0)$. *The readers should have no trouble to walk through the evolution at time 7 and at time 8.*

3.3 The proof

We now prove that with Algorithm 1, the construction in Figure 3 behaves like a FIFO queue with buffer $2^{k+1} - 1$ when it is started from the empty system.

Lemma 5. *If packet* p *is on the active line at some time* t *and moved forward to another cell (or out of the system), then packet* $p + 1$ *is on the active line at time* $t + 1$.

Proof. Suppose that packet p is stored in $b_i(j)$ during the previous move of packet $p+1$. Then the ideal buffer of packet $p+1$ in it previous move is $b_i(j+1)$. Since $b_i(j)$ is assumed to be on the active line at time t, $b_i(j+1)$ is on the active line at time $t+1$. In view of Lemma 4, we need to consider the following two cases.
Case 1. packet $p+1$ is stored in its target buffer in its previous move:

From Corollary 1, it follows that the target buffer of packet $p+1$ in it previous move is on every path from $b_{2k}(0)$ to $b_i(j+1)$. As packet $p+1$ is stored in its target buffer in its previous move and $b_i(j+1)$ is on the active line at time $t+1$, packet $p+1$ is also on the active line at time $t+1$.
Case 2. packet $p+1$ is stored in a predecessor buffer of its ideal buffer in its previous move:

From Lemma 4 and (P6), we know (i) packet $p+1$ is stored in a predecessor buffer of $b_i(j+1)$, and (ii) cell i is full and contains packets from $p - 2^i + 1$ to p with packet $p - 2^i + 1$ in $b_i(j+1)$. For $i \geq k$, $b_i(j+1)$ only has one predecessor buffer. Thus, both $b_i(j+1)$ and its predecessor buffer are on the active line at time $t+1$. For $i < k$, there are two predecessor buffers of $b_i(j+1)$. One is active at time $t+1$ and the other is active at time $t - 2^i + 1$. If packet $p+1$ is stored in the one that is active at time $t+1$, then the proof is completed. Thus, it suffices to consider the case that packet $p+1$ is stored in the predecessor buffer that is active at time $t - 2^i + 1$. Since packet p is assumed to be moved to another cell at time t, packet $p - 2^i + 1$, as the first packet of cell i, must have been be moved to another cell not later than time $t - 2^i + 1$ (because it takes at least one time slot to move a packet). As such, packet $p+1$ can be put in $b_i(j+1)$ at time $t - 2^i + 1$ and it is then on the active line at time $t+1$. □

Lemma 6. *The first packet on the active line is always the next packet to serve. More precisely, if packet p is moved to the output link at time t, then at time $t+1$, packet $p+1$ can be either moved to the output link (if $c(t+1)=1$) or be moved to buffer $b_0(0)$ (if $c(t+1)=0$).*

Proof. We prove the result by induction on the number of arrived packets in the system. When no packet entered the system yet, the result trivially holds. Suppose now that for the first p packets, that result holds. Now consider packet $p+1$. Suppose that packet p is moved to the output link at time t. Then it follows from Lemma 5 that packet $p+1$ is on the active line at time $t+1$. As such, packet $p+1$ can be either moved to the output link (if $c(t+1)=1$) or be moved to buffer $b_0(0)$ (if $c(t+1)=0$). □

Lemma 6 shows that the non-idling property in (P2) and the FIFO property in (P4) hold. It remains to show that the capacity of that queue is at least $2^{k+1}-1$. For this, we use an argument similar to that in [2]. During one busy period (and let say that it begins at time 0), the odd lines between cells $2k-1$ and 1 form a FIFO queue for the odd time slots and the even lines between cells $2k-1$ and 1 form another FIFO queue for the even time slots. Indeed, a packet can enter a buffer on a even line between cells $2k-1$ and 1 at even time slots, and move on it and exit it at even time slots. The same holds for the odd lines at odd time slots. As the target buffer of a packet is always on the next line of that of the previous packet, every two arriving packet is sent to one of the two queues while the other packets are sent to the other. We call the system composed by the odd lines the *odd queue* and the system composed by the even lines the *even queue*.

Lemma 7. *Let n_o (resp. n_e) be the number of packets in the odd (resp. even) queue. Then $|n_o - n_e| \leq 1$ and if the next packet must be routed to the odd (resp. even) queue, then $n_e \geq n_o$ (resp. $n_o \geq n_e$).*

Proof. Suppose, without loss of generality, that during a busy period, odd (resp. even) numbered packets are sent to the odd (resp. even) queue and that the first packet in the buffer is an odd numbered packet. As the queue is FIFO, if there are p packets in the system, they are numbered from $2m+1$ to $2m+1+p-1$ (for some m) and $0 \leq n_o - n_e \leq 1$. If the next packet to arrive is odd, then this means that $n_o = n_e$ and if the next packet to arrive is even then $n_o - n_e = 1$. □

Lemma 8. *If there are less than $2^{k+1}-1$ packets in the system, then the system can accept new packets.*

Proof. We show the result by induction on k. For $k=1$, the queue composed of one cell of delay 1 is clearly a FIFO queue with buffer 1. Suppose that the queue composed of $2k-1$ optical cells can accept at least 2^k-1 packets. Consider a system composed of $2k+1$ optical cells. Its odd and even queues can accept at least 2^k-1 packets.

If a packet cannot be accepted at time t, this is because the last optical cell, cell $2k$, is occupied and cannot be moved. That cell was also occupied with the same packet at time $t-1$. This means that the queue where the packet has to be routed contains at least 2^k-1 packets and, according to Lemma 7, the other queue also contains at least 2^k-1 packets. In the whole, the number of packets in the system is at least $2(2^k-1)+1 = 2^{k+1}-1$. □

The previous lemmas enable us to state the following theorem.

Theorem 1. *With the algorithm described above, the system of Fig. 3, when it starts from the empty system, is a FIFO queue with buffer size $2^{k+1}-1$.*

Proof. Lemma 6 ensures that (P2) and (P4) are satisfied and Lemma 8 ensures that (P3) is satisfied. (P1) is also satisfied: a packet is routed outside of the system if and only if there is a departure (lines 3-6); if there is an arrival, then, a new packet is routed into the system (lines 12-16); and in between, packets are just moved inside the system (lines 7-11). □

3.4 Complexity issues

We now analyze the complexity of the algorithm for a FIFO queue with buffer $B = 2^{k+1}-1$. As the packets are ordered by the index of cells and they are contiguous in every cell, we only have to keep track of the first and last occupied buffers in each cell. In a whole, we need to keep in memory $O(\log B)$ numbers ranging from 0 to 2^k-1. As a consequence, the space complexity is in $O((\log B)^2)$.

Each cell in the system is visited one time at each time slot. For each packet that can be moved, one needs to find its target cell, and update the first and last occupied buffer in the cells. Those operations can be done in $O(\log B)$ time. So the time complexity is also in $O((\log B)^2)$.

Number of optical cells	$O(\log B)$
Space complexity	$O((\log B)^2)$
Time complexity	$O((\log B)^2)$

4 Conclusion

In this paper, we provided an explicit control algorithm for the optical FIFO queues constructed in [2]. One of the questions that remains unanswered is whether the construction in [2] is optimal in terms of the number of 2×2 switches. We note that the construction in Figure 3 can accept at most $2^{k+1} - 1$ packets as a FIFO queue. To see this, consider the queue of Fig. 9. If there are 4 packets in the queue, a new packet cannot be accepted when line 1 is active, even if there is a departure. Thus, the buffer of the FIFO queue cannot exceed 3.

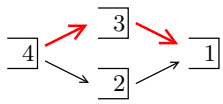

Figure 9: Buffer of size 3 only.

References

[1] C.-S. Chang, Y.-T. Chen, J. Cheng, and D.-S. Lee. Multistage constructions of linear compressors, non-overtaking delay lines, and flexible delay lines. In *Proceedings of IEEE INFOCOM*, 2006.

[2] C.-S. Chang, Y.-T. Chen, and D.-S. Lee. Constructions of optical FIFO queues. *IEEE Transactions on Information Theory*, 52(6):2838–2843, 2006.

[3] Cheng-Shang Chang, Duan-Shin Lee, and Chao-Kai Tu. Recursive construction of FIFO optical multiplexers with switched delay lines. *IEEE Transactions on Information Theory*, 50(12):3221–3233, 2004.

[4] Imrich Chlamtac, Andrea Fumagalli, Leonid Kazovsky, Paul Melman, William H. Nelson, Pierluigi Poggiolini, Mauro Cerisola, A. N. M. Masum Choudhury, Thomas K. Fong, R. Theodore Hofmeister, Chung-Li Lu, Adisak Mekkittikul, Delfin Jay M. Sabido IX, Chang-Jin Suh, and Eric W. M. Wong. CORD: Contention resolution by delay lines. *IEEE Journal on Selected Areas in Communications*, 14(5):1014–1029, 1996.

[5] C.-C. Chou, C.-S. Chang, D.-S. Lee, and J. Cheng. A necessary and sufficient condition for the construction of 2-to-1 optical fifo multiplexers by a single crossbar switch and fiber delay lines. *EEE Transactions on Information Theory*, 52:4519–4531, 2006.

[6] Rene L. Cruz and Jung-Tsung Tsai. COD: alternative architectures for high speed packet switching. *IEEE/ACM Transactions on Networking*, 4(1):11–21, 1996.

[7] D. Hunter, M. Chia, and I. Andonovic. Buffering in optical packet switches.

[8] S. Iyer and N. McKeown. Making parallel packet switches practical. In *INFOCOM*, pages 1680–1687, 2001.

[9] Anand D. Sarwate and Venkat Anantharam. Exact emulation of a priority queue with a switch and delay lines. *Queueing Syst.*, 53(3):115–125, 2006.

[10] S. Yao, B. Mukherjer, and S. Dixit. Advances in photonic packet switching: An overview. *IEEE Communications Magazine*, 38:84–94, february 2000.

Unité de recherche INRIA Rennes
IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)

Unité de recherche INRIA Futurs : Parc Club Orsay Université - ZAC des Vignes
4, rue Jacques Monod - 91893 ORSAY Cedex (France)
Unité de recherche INRIA Lorraine : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)
Unité de recherche INRIA Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)
Unité de recherche INRIA Sophia Antipolis : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399