Theoretical Computer Science 409 (2008) 477–485

Contents lists available at ScienceDirect

Theoretical Computer Science

journal homepage: www.elsevier.com/locate/tcs

Deadline guaranteed packet scheduling for overloaded traffic in input-queued switches

Xiaojun Shen [a,*], Jianyu Lou [a], Weifa Liang [b], Junzhou Luo [c]

[a] School of Computing and Engineering, University of Missouri-Kansas City, 5100 Rockhill Road, Kansas City, MO 64110, USA

[b] Department of Computer Science, Australian National University, Canberra, ACT 0200, Australia

[c] School of Computer Science and Engineering, Southeast University, Nanjing 210096, PR China

ARTICLE INFO

Article history:
Received 7 May 2008
Received in revised form 26 August 2008
Accepted 5 September 2008
Communicated by D.-Z. Du

Keywords:
Input-queued switch
Packet scheduling
Quality of service
Deadline guarantee
NP-hard

ABSTRACT

Many applications need to solve the deadline guaranteed packet scheduling problem. However, it is a very difficult problem if three or more deadlines are present in a set of packets to be scheduled. The traditional approach to dealing with this problem is to use EDF (Earliest Deadline First) or similar methods. Recently, a non-EDF based algorithm was proposed that constantly produces a higher throughput than EDF-based algorithms by repeatedly finding an optimal scheduling for two classes. However, this new method requires the two classes be non-overloaded, which greatly restricts its applications. Since the overloaded situation is not avoidable from one iteration to the next in dealing with multiple classes, it is compelling to answer the open question: Can we find an optimal schedule for two overloaded classes efficiently? This paper first proves that this problem is NP-complete. Then, this paper proposes an optimal preprocessing algorithm that guarantees to drop a minimum number of packets from the two classes such that the remaining set is non-overloaded. This result directly improves on the new method.

1. Introduction

Because traditional output queued switches have poor scalability, input queuing (IQ) schemes become widely used switching architecture [2–8]. In an input-queued switch, incoming packets are first queued on the input side, and then scheduled so that they will be transmitted to destined output ports without output contention. A number of scheduling algorithms have been proposed that are based on various maximum weighted matching (MWM), or maximal matching [3–6]. They compute a (maximum or maximal) matching for each time slot. Since a matching pairs each input port to a unique output port and vice versa, a packet from an input port can be transmitted to its matched output port without contention in a scheduled slot.

Although some of these existing algorithms provide 100% throughput statistically [3–5], they do not provide deterministic packet delay guarantees, which affects advanced applications. Moreover, they need to compute the matching for every time slot, independently. As the slot time becomes shorter and shorter due to increasing line speed, a scalability problem arises.

To reduce the time complexity, the frame-based packet scheduling has been proposed [9,10], where a frame consists of a fixed number of consecutive time slots. The scheduling algorithm fetches the arriving packets buffered in the input side once for each frame instead of each slot. When all packets in the current cycle (frame) are scheduled, the scheduling algorithm fetches the packets in the buffer received during the current cycle and repeats the computation again. Depending on the

* Corresponding author. Tel.: +1 816 235 2223.
E-mail addresses: shenx@umkc.edu (X. Shen), jlad2@umkc.edu (J. Lou), wliang@cs.anu.edu.au (W. Liang), jluo@seu.edu.cn (J. Luo).

0304-3975/$ – see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2008.09.013

size of the frame, the algorithm could produce schedules for many time slots in one execution. Since the frame length can be designed according to the complexity of the algorithm, the frame-based scheduling effectively reduces the computational complexity.

In order to provide guaranteed delay performance for IQ switches, we consider the following frame-based scheduling problem: Given a set of packets each of which has a deadline, can we find a feasible contention-free schedule such that every packet can be transferred to its destined output port before its deadline? If no such a feasible schedule exists, how can we find a schedule that allows a maximum number of packets to meet their deadlines? These two related questions are called the *schedulability* problem and the *optimization* problem, respectively [1]. Since deadline guarantee implies throughput and rate guarantees, they have also been hot problems studied in the fields of multi-periodic satellite switches (SS/TDMA) scheduling, optical network scheduling and real-time scheduling, etc. [12–15]. The deadlines for packets in these systems are either pre-specified or determined by the current period [12,13], or derived from traffic requirements [14]. Paper [16] elaborates how to compute the virtual time stamps (expected arrival and departing times) from the end-to-end delay requirement and insert them into packets.

For the simplest case, where all packets have the same deadline, we can apply the Birkhoff-Von Neumann algorithm to decompose the traffic matrix into a linear combination of permutation matrices, each corresponding to a contention-free schedule for one time slot [11]. However, when three or more deadlines are present in the packet set, the scheduling problem becomes NP-complete [12]. Traditionally, EDF (Earliest Deadline First) algorithm or its variants are used to schedule the set. Paper [13] provides a good survey of these algorithms.

Recently, we proposed a non-EDF based algorithm [1] called *Flow-based Iterative Packet Scheduling* (FIPS). A key component of this algorithm is a procedure that produces an optimal schedule for a packet set with two classes (deadlines). An optimal schedule is one that allows maximum number of packets to be transmitted before their deadlines. FIPS solves the multi-class scheduling problem by repeatedly applying the procedure for two classes. Although this method seems very promising, the existing procedure for two classes is only applicable when the traffic of the two classes is non-overloaded (defined in Theorem 1, Section 2). Since the procedure may create an overloaded situation from one iteration to the next (even all classes are non-overloaded initially), it is very much desired to know: Can we find an optimal schedule efficiently for two overloaded classes?

This paper proves that the above optimization problem is NP-complete if both classes are overloaded. In order to deal with the NP-completeness, this paper proposes a preprocessing algorithm that guarantees to drop a minimum number of packets such that the remaining set is non-overloaded. This result directly improves on the current FIPS algorithm.

The rest of this paper is organized as follows. Section 2 introduces necessary notations and background to the deadline guaranteed packet-scheduling problem. Section 3 proves the optimal scheduling problem for two overloaded classes is NP-complete. Section 4 presents an optimal preprocessing algorithm and Section 5 concludes this paper.

2. Preliminaries

Consider an $N \times N$ input-queued switch. We assume that all packets have a fixed size. (Variable lengths are dealt with by the segmentation and reassembly technique.) Accordingly, the time is divided into slots of a fixed length that equals to the transmission time of a packet. Moreover, let T consecutive slots constitute a frame. In each time slot, according to a schedule, the switch controller establishes N one-to-one connections from input ports to output ports that allow at most one packet be transferred from each input port to the paired output port. Slots in each frame are numbered from 0. Slot k covers time interval $[t_k, t_{k+1})$. A triple (i, j, d) is associated with a packet that is to be transmitted from input port i to output port j with deadline t_d. It is possible that multiple packets have the same triple (i, j, d). If a packet of (i, j, d) is scheduled in slot k, $0 \le k \le d$, we say that *it is deadline guaranteed.*

Let S be a set of packets to be scheduled. For each fixed i, j, and d, $1 \le i, j \le N$, let $S^d_{i,j} = \{\text{all packets of triple } (i, j, d)\}$. Also, let $p^d_{i,j} = \left|S^d_{i,j}\right|$, $p^d_i = \sum_{j=1}^N p^d_{i,j}$, and $q^d_j = \sum_{i=1}^N p^d_{i,j}$.

Our goal is to find a schedule that allows a maximum number of packets to meet their deadlines. Packets that miss their deadlines would be dropped. If a dropped packet has a deadline greater than T, it will join the next frame with its deadline reduced by T from the current one.

Let $S_{i,j}(k) = \{\text{packets of } (i, j, d) \text{ that are scheduled in slot } k\}$ and $p_{i,j}(k) = \left|S_{i,j}(k)\right|$.

Since at most one packet can be transferred between an input port and an output port in any time slot, for any k, the following conditions must be satisfied

$$\sum_{j=1}^{N} p_{i,j}(k) \le 1, \quad i = 1, \ldots, N, \quad \text{and}$$

$$\sum_{i=1}^{N} p_{i,j}(k) \le 1, \quad j = 1, \ldots, N. \tag{1}$$

Definition 1. A packet set S is called *schedulable* if a schedule exists such that conditions of (1) are satisfied and every packet is deadline guaranteed.

Definition 2. The *schedulability problem* is to determine whether a given packet set S is schedulable.

Let D be the set of distinct deadlines that occur in packet set S, $D = \{d_1, d_2, d_3, \ldots, d_k\}$, $0 < d_1 < d_2 < d_3 < \cdots < d_k$. The set of all packets with deadline d_i is called class i, $i = 1, 2, \ldots, k$.

A necessary condition for set S to be schedulable is called *non-overloaded* conditions [1,11], which is stated in Theorem 1.

Theorem 1. *Let S be a set of packets with distinct deadlines $d_1 < d_2 < d_3 < \cdots < d_k$. The set S is schedulable only if the following non-overloaded conditions are satisfied:*

$$\sum_{l=1}^{m} p_i^{d_l} \le d_m + 1 \quad \text{and} \quad \sum_{l=1}^{m} q_j^{d_l} \le d_m + 1 \quad \text{for any } i, j, \text{ and } m,\ 1 \le i, j \le N,\ 1 \le m \le k. \tag{2}$$

Theorem 1 is obviously true because the number of packets arriving at any specific input port i (or destined to any specific output port j) with deadlines less than or equal to d_m cannot be larger than the number of time slots available which is $d_m + 1$.

If set S is not schedulable, we wish to find a schedule that guarantees a maximum number of packets to meet their deadlines. We define a related optimization problem.

Definition 3. Given a packet set S, an *optimal schedule without priority structure* is one that allows the maximum number of packets to meet their deadlines, or equivalently, it is one that drops the minimum number of packets such that the remaining packets are schedulable.

Definition 4. Let S be a packet set with distinct deadlines $d_1 < d_2 < d_3 < \cdots < d_k$. Let R be a feasible schedule after dropping n_i packets of class i, $i = 1, 2, \ldots, k$. R is called an *optimal schedule with priority structure* if the sequence $n_1, n_2, \ldots, n_k$ is lexicographically the smallest.

When S is schedulable, there is an optimal schedule that guarantees all packets to meet their deadlines without packet dropping. It is proved that the schedulability problem is NP-complete when $|D| \ge 3$. When $|D| \le 2$, some cases are solved and some remains open. We summarize the results as follows.

A. *The case $|D| = 1$.*

In this case, the necessary condition (2) becomes a sufficient condition also [11]. We state it in Theorem 2.

Theorem 2. *A packet set S is schedulable, where all packets have a common deadline d, if and only if $(p_i^d \le d+1)$ and $(q_j^d \le d+1)$ for all i, j, $1 \le i, j \le N$.*

When condition (2) is satisfied, an optimal schedule can be found efficiently [1].

B. *The case $|D| = 2$.*

For this case, paper [1] showed an efficient algorithm for solving the schedulability problem and the optimization problem with priority structure when the traffic is non-overloaded. An open question is whether the optimization problem is polynomial solvable when the traffic is overloaded.

3. NP-completeness of the optimization problem for two classes

Suppose two deadlines, $d_1 < d_2$, are present in a packet set S that is to be scheduled on an $N \times N$ input-queued switch. Let $A = \{\text{packets with deadline } d_1\}$, $B = \{\text{packets with deadline } d_2\}$, and $S = A \cup B$. From Theorem 1, the non-overloaded conditions are:

$$p_i^{d_1} \le d_1 + 1 \quad (i = 1, \ldots, N) \quad \text{and} \quad q_j^{d_1} \le d_1 + 1 \quad (j = 1, \ldots, N) \tag{3}$$

and

$$p_i^{d_1} + p_i^{d_2} \le d_2 + 1 \quad (i = 1, \ldots, N) \quad \text{and} \quad q_j^{d_1} + q_j^{d_2} \le d_2 + 1 \quad (j = 1, \ldots, N). \tag{4}$$

We will show that if neither condition (3) nor condition (4) is satisfied, then the optimal scheduling problem is NP-complete. For convenience, class 1 and class 2 are also called class A and class B (or set A and set B), respectively. Before we present the proof, we need to define a corresponding decision problem.

Definition 5. Let S be a packet set as defined above. The *two-class dropping problem with priority structure* is to determine whether S can be scheduled by dropping at most p_1 packets of class A and at most p_2 packets of class B. We denote this problem by *Two-Class-Dropping-with-Priority*(A, B, p_1, p_2).

Definition 6. For set A, we define the *excess degree* for each input port i and each output port j to be $a_i^{d_1} = p_i^{d_1} - (d_1 + 1)$ and $b_j^{d_1} = q_j^{d_1} - (d_1 + 1)$ for all i, j, $1 \le i, j \le N$, respectively.

Definition 7. For set A, we define the *vacancy degree* for each input port i and each output port j to be $c_i^{d_1} = d_1 + 1 - p_i^{d_1}$ and $e_j^{d_1} = d_1 + 1 - q_j^{d_1}$ for all i, j, $1 \le i, j \le N$, respectively.

A =

	1	2	3	4	5	$p_i^{d_1}$	$a_i^{d_1}$	$c_i^{d_1}$
1	3	0	1	0	0	4	1	-1
2	0	0	3	0	1	4	1	-1
3	0	1	0	3	0	4	1	-1
4	0	0	0	0	1	1	-2	2
5	0	2	0	0	0	2	-1	1
$q_j^{d_1}$	3	3	4	3	2			
$b_j^{d_1}$	0	0	1	0	-1			
$e_j^{d_1}$	0	0	-1	0	1			

(a)

B =

	1	2	3	4	5	$p_i^{d_2}$	$a_i^{d_2}$
1	0	3	0	0	4	7	5
2	2	0	1	0	0	3	1
3	0	2	0	1	0	3	1
4	1	0	0	0	1	2	0
5	0	0	1	2	0	3	1
$q_j^{d_2}$	3	5	2	3	5		
$b_j^{d_2}$	1	3	0	1	3		

(b)

Fig. 1. Vacancy degrees and excess degrees for set A with $d_1 = 2$ and set B with $d_2 = 4$.

Fig. 1(a) shows an example of set A for a 5×5 switch with $d_1 = 2$. The number at entry (i, j) is the number of class A packets from input port i to output port $j(= p_{i,j}^{d_1})$, for all i, j, $1 \leq i, j \leq N$. The vacancy degree and excess degree for each input and output port are also given in Fig. 1(a). Note that $c_i^{d_1} = -a_i^{d_1}$ and $e_j^{d_1} = -b_j^{d_1}$ for all i, j, $1 \leq i, j \leq N$.

Obviously, if an input (or output) port has a positive excess degree p, then at least p class A packets from (or to) this port must be dropped.

Definition 8. For set B, we define the *excess degree* for each input port i and each output port j to be $a_i^{d_2} = p_i^{d_2} - (d_2 - d_1)$ and $b_j^{d_2} = q_j^{d_2} - (d_2 - d_1)$ for all i, j, $1 \leq i, j \leq N$, respectively.

Fig. 1(b) shows an example for set B, where $d_2 = 4$.

If an input (or output) port has a positive excess degree p in set B, then at least p class B packets from (or to) this port must be scheduled before time d_1 or dropped. Those class B packets scheduled before time d_1 are said to be promoted to class A.

Theorem 3. *The Two-Class-Dropping-with-Priority(A, B, p_1, p_2) problem is NP-complete.*

Proof. Since this problem is clearly in NP class, we only need to show it is NP-hard. In the following, we show how to reduce the 3-SAT problem [18] to this scheduling problem. Consider a Boolean formula ϕ that contains n Boolean variables, $y_1, y_2, \ldots, y_n$. We assume that ϕ is in 3-CNF (Conjunctive Normal Form) which consists of m clauses connected by *and* ($\wedge$) operators, $\phi = C_1 \wedge C_2 \wedge \cdots \wedge C_m$, where each C_i ($1 \leq i \leq m$) consists of exactly three distinct literals connected by *or* ($\vee$) operators. Given a 3-CNF formula, the 3-SAT problem is: Does there exist an assignment of the n variables such that the truth value of the formula is true? If yes, we say that the 3-CNF is satisfiable.

Given a 3-CNF formula with n variables and m clauses, we construct a packet set S for the *Two-Class-Dropping-with-Priority*(A, B, p_1, p_2) problem such that the 3-CNF formula is satisfiable if and only if the answer to the *Two-Class-Dropping-with-Priority*(A, B, p_1, p_2) problem is true.

In our construction, we use $d_1 = 0$ and $d_2 = 2$. Details are given by the following two steps.

Step 1 For each variable $x \in \{y_1, y_2, \ldots, y_n\}$, construct a set of class A packets and class B packets as follows.

Suppose x occurs k times in ϕ and $\bar{x}$ occurs l times. Without loss of generality, let $k \geq l$. The set of packets constructed for these literals is illustrated by Fig. 2. We use x_i ($1 \leq i \leq k$) to represent the ith occurrence of x and $\overline{x_j}$ ($1 \leq j \leq l$) to represent the jth occurrence of $\bar{x}$.

In Fig. 2, each edge from an input port to an output port represents a packet. So, there are $4k$ class A packets and $7k$ class B packets related to variable x. Each of labels b_i, x_i, u_i, v_i, $\overline{x_i}$, $1 \leq i \leq k$, represents a distinct port. Literal x_i corresponds to input port x_i ($1 \leq i \leq k$) and literal $\overline{x_i}$ corresponds to input port $\overline{x_i}$ ($1 \leq i \leq l$). Because $l \leq k$, ports $\overline{x_i}$ ($l < i \leq k$) have no corresponding literals. If $l > k$, the construction can be done similarly, So, we assume $l \leq k$ for every variable. Let us make some observations.

Because $d_1 = 0$, either packet $(b_i, x_i, 0)$ or $(b_i, \overline{x_i}, 0)$ in class A must be dropped, $1 \leq i \leq k$. Because of the priority structure, we must drop exactly k packets from the $4k$ class A packets. Otherwise, the schedule cannot be optimal. In addition, because input ports u_i ($1 \leq i \leq k$) have an excess degree in class B but no vacancy degree in class A, we must drop one class B packet from input port u_i ($1 \leq i \leq k$). Therefore, at least we need to drop k class B packets.

Let variable y_j ($1 \leq j \leq n$) occurs k_j times in formula ϕ and $\overline{y_j}$ occurs l_j times, $k_j \geq l_j$. Then, we need to drop at least $p_1 = \sum_{j=1}^{n} k_j$ class A packets and the same number of $p_2 = p_1 = \sum_{j=1}^{n} k_j$ class B packets to make the remaining set schedulable.

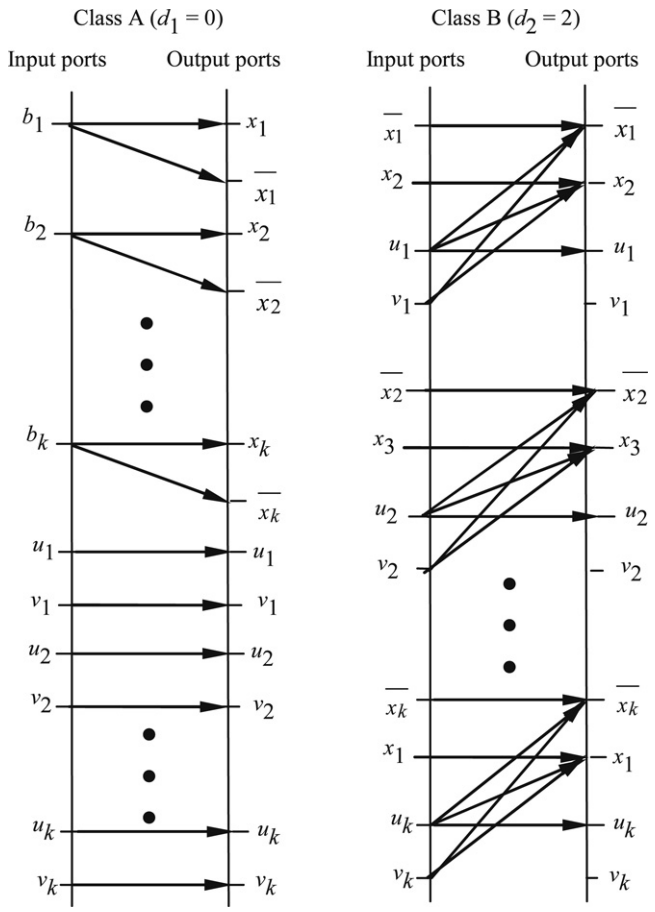


Fig. 2. The set of packets for variable *x*.

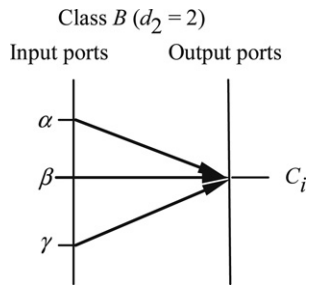


Fig. 3. Three more class *B* packets are added for each clause.

Step 2 For each clause C_i $(1 \leq i \leq m)$, we construct three more class *B* packets as follows. Let α, β,γ be the three literals contained in C_i. We add three more class *B* packets as illustrated in Fig. 3, where C_i is a distinct port.

Now, we prove that the 3-CNF formula ϕ is satisfiable if and only if we can schedule the packets constructed above by dropping $p_1 = \sum_{j=1}^{n} k_j$ class *A* packets and the same number of $p_2 = p_1 = \sum_{j=1}^{n} k_j$ class *B* packets.

(i) Suppose formula ϕ is satisfiable. We schedule the packets in the following way.

We check each variable $x \in \{y_1, y_2, \ldots, y_n\}$. If x is assigned value true and occurs k times in ϕ, then all its k occurrences must be assigned true and all occurrences of $\bar{x}$ must be false. Thus, we drop all packets of $(b_i, \overline{x_i}, 0)$ $(1 \leq i \leq k)$ from class *A*. The remaining class *A* packets are obviously schedulable. Next, we promote all packets of $(\overline{x_i}, \overline{x_i}, 2)$ $(1 \leq i \leq k)$ from class *B* to class *A*. Because $(b_i, \overline{x_i}, 0)$ has been dropped, the vacant output ports $\overline{x_i}$ can be used for $(\overline{x_i}, \overline{x_i}, 2)$ $(1 \leq i \leq k)$. Third, we drop all packets of $(u_i, x_i, 2)$ $(1 \leq i \leq k)$ from class *B*. Now, all remaining packets constructed in Step 1 are schedulable. The case where x is assigned false is just a symmetric case. We omit details.

Now, we check each clause C_i $(1 \leq i \leq m)$. Let α, β,γ be the three literals contained in C_i. One of them must be assigned true, say α is assigned true. Then we promote the packet $(\alpha, C_i, 2)$ to class *A*. Because α is assigned true,

$(\alpha, \alpha, 2)$ is not promoted. So, input ports α and output port C_i are available in slot 0 for $(\alpha, C_i, 2)$. Obviously, the two remaining packets constructed in Step 2 are schedulable after this promotion. Therefore, we have proved that if formula ϕ is satisfiable, then the packet set we constructed are schedulable by dropping $p_1 = \sum_{j=1}^{n} k_j$ class A packets and the same number of class B packets.

(ii) Suppose that the packet set we constructed is schedulable by dropping $p_1 = \sum_{j=1}^{n} k_j$ class A packets and the same number of class B packets. We shall show that the formula is satisfiable.

From the observations made in Step 1, for variable y_j, we must drop k_j class A packets of $(b_i, x_i, 0)$ or $(b_i, \overline{x_i}, 0)$, $(1 \le i \le k_j)$. We also must drop one class B packet from input port u_i $(1 \le i \le k_j)$. Because class B packets have excess degree one at each output port x_i and $\overline{x_i}$, in order not to drop more packets, either $(u_i, \overline{x_i}, 2)$ or $(u_i, x_{i+1}, 2)$ must be dropped $(1 \le i \le k_j)$, where $i+1$ is assumed to be $(i+1)\ mod\ k_j$, so that $(i+1)\ mod\ k_j = 1$, when $i = k_j$.

Consider the set U of packets (or edges) we discussed above for possible dropping.

$$U = \{(b_i, x_i, 0) \text{ and } (b_i, \overline{x_i}, 0) | (1 \le i \le k_j)\} \cup \{(u_i, \overline{x_i}, 2) \text{ and } (u_i, x_{i+1}, 2) | (1 \le i \le k_j)\}.$$

Since all edges in set U form a single cycle, we must drop all packets of $(b_i, x_i, 0)$ or all packets of $(b_i, \overline{x_i}, 0)$. If we drop all packets of $(b_i, x_i, 0)$, then we must promote all packets of $(x_i, x_i, 2)$ and drop all packets of $(u_i, \overline{x_i}, 2)$, $(1 \le i \le k_j)$. If we drop all packets of $(b_i, \overline{x_i}, 0)$, then we must promote all packets of $(\overline{x_i}, \overline{x_i}, 2)$ and drop all packets of $(u_i, x_{i+1}, 2)$, $(1 \le i \le k_j)$. This means that if the packets constructed in Step 1 are schedulable after dropping $p_1 = \sum_{j=1}^{n} k_j$ class A packets and the same number of class B packets, then any feasible schedule must leave all input ports x_i open in time slot 0 or all input ports $\overline{x_i}$ open, but not both. These two situations correspond to the two Boolean values of y_j. If all ports x_i open in time slot 0 and y_j occurs in clause C_i, then the packet $(y_j, C_i, 2)$ constructed in Step 2 for this occurrence can be promoted to class A. However, any packet of $(\overline{y_j}, C_i, 2)$ cannot be promoted. Because the packets set is schedulable by dropping $p_1 = \sum_{j=1}^{n} k_j$ class A packets and the same number of class B packets, at least one packet from clause C_i is promoted to class A. Therefore, we assign true to literal α if packet $(\alpha, C_i, 2)$ is promoted to class A. This is a valid assignment because, for any literal α, a packet of $(\alpha, C_i, 2)$ and a packet of $(\overline{\alpha}, C_j, 2)$ cannot both be promoted. Moreover, this assignment satisfies formula ϕ because at least one literal in each clause is assigned true. □

Now, we prove that the two-class dropping problem without priority structure is also NP-complete.

Definition 9. Let S be the same packet set as that in Definition 5. The *two-class dropping problem without priority structure* is to determine whether S can be scheduled by dropping at most p packets that can belong to class A or class B. We denote this problem by *Two-Class-Dropping-without-Priority*(A, B, p).

Theorem 4. *The Two-Class-Dropping-without-Priority(A, B, p) problem is NP-complete.*

Proof. The proof is almost the same as that for Theorem 3. We reduce the 3-SAT problem to this scheduling problem also. Given a 3-CNF formula ϕ with n variables and m clauses, we construct a packet set S in exactly the same way as we did in Theorem 3. Now we prove that formula ϕ is satisfiable if and only if the set S is schedulable by dropping a total of $p = 2\sum_{j=1}^{n} k_j$ packets of class A or B.

(i) Suppose formula ϕ is satisfiable. Then we can drop $p_1 = \sum_{j=1}^{n} k_j$ class A packets and the same number of class B packets as we did in the proof of Theorem 3 such that the remaining set is schedulable. Obviously, the set S is schedulable by dropping a total of $p = 2\sum_{j=1}^{n} k_j$ packets of class A or B.

(ii) Suppose that the packet set S is schedulable by dropping $p = 2\sum_{j=1}^{n} k_j$ class A or class B packets. We shall show that the formula ϕ is satisfiable. We notice that to make S schedulable, among the packets constructed for variable y_j, $(1 \le j \le n)$, we must drop at least the following packets:

(a) packet $(b_i, x_i, 0)$ or $(b_i, \overline{x_i}, 0)$ in class $A (1 \le i \le k_j)$.

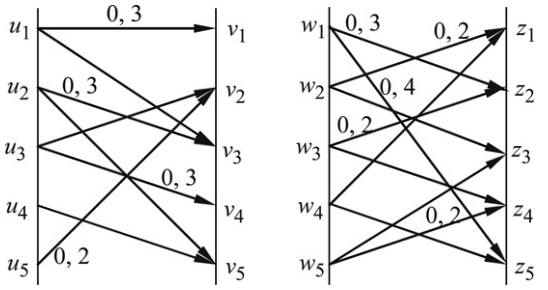
(b) one packet whose input port is u_i, $(1 \le i \le k_j)$. That is, one of the four packets of $(u_i, u_i, 0)$, $(u_i, \overline{x_i}, 2)$, $(u_i, x_{i+1}, 2)$, $(u_i, u_i, 2)$ must be dropped $(1 \le i \le k_j)$. This is because the total number of available slots is three and there are four packets at this port, one class A packet and three class B packets. Without priority structure, we allow to drop $(u_i, u_i, 0)$ which is a class A packet.

Since the total number of packets we must drop in (a) and (b) is at least $p = 2\sum_{j=1}^{n} k_j$, we must drop exactly one of $(b_i, x_i, 0)$ or $(b_i, \overline{x_i}, 0)$ in class A. Since we also have one excess degree at output pots x_i and $\overline{x_i}$ $(1 \le i \le k_j)$ in class B, in order not to drop more packets, we must promote k_j class B packets from these ports and drop k_j class B packets from these ports, which implies that we must drop k_j class B packets of $(u_i, \overline{x_i}, 2)$ or $(u_i, x_{i+1}, 2)$ $(1 \le i \le k_j)$ in part (b).

Therefore, we conclude that:

(1) All packets of $(b_i, x_i, 0)$ (or all packets of $(b_i, \overline{x_i}, 0)$) in class $A (1 \le i \le k_j)$ are dropped, but not both.
(2) Correspondingly, all packets of $(u_i, \overline{x_i}, 2)$ (or all packets of $(u_i, x_{i+1}, 2)$) $(1 \le i \le k_j)$ are dropped.
(3) All packets of $(x_i, x_i, 2)$ (or all packets of $(\overline{x_i}, \overline{x_i}, 2)$) $(1 \le i \le k_j)$ are promoted.

Then the remaining proof can be done in exactly the same way we did for Theorem 3. □



(a) The bipartite graph for set A. (b) The bipartite graph for set B.

Fig. 4. An illustration of bipartite graphs for set A and set B.

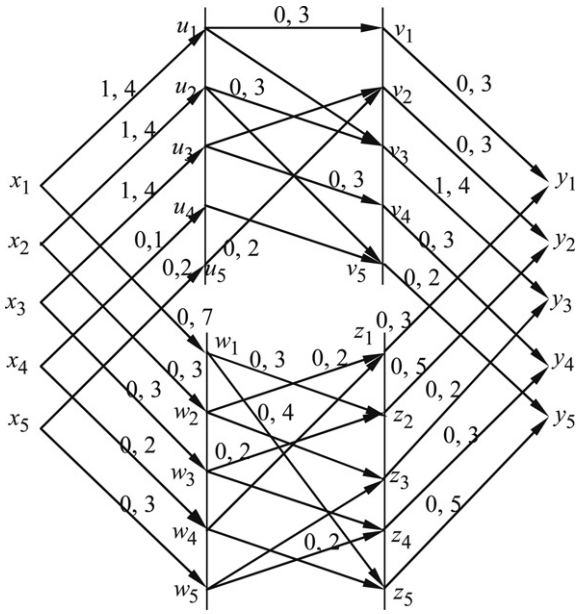


Fig. 5. An illustration of Step 4.

4. An optimal preprocessing algorithm

In order to deal with the NP-hardness of the two-class scheduling problem when the traffic is overloaded, we propose a preprocessing algorithm that drops a minimum number of packets such that the remaining set is non-overloaded. Given traffic matrices A and B as we defined in Section 3, we will transform them to a network flow problem, where every edge in the network is labeled with two non-negative integers (α, β), where α is called a lower bound and β an upper bound. The following procedure shows the steps for the transformation. We use the example in Fig. 1 to illustrate.

Procedure Network-Graph (A, B, G)

Step 1 Compute excess degrees $a_i^{d_1}$ and $b_j^{d_1}$ for set A and excess degrees $a_i^{d_2}$ and $b_j^{d_2}$ for set $B (1 \le i \le N)$.

Step 2 Construct a bipartite graph $G_1(U_1, V_1, E_1)$ for set A, where $U_1 = \{u_1, u_2, \ldots, u_N\}$ are N vertices corresponding to N input ports, $V_1 = \{v_1, v_2, \ldots, v_N\}$ are N vertices corresponding to N output ports. An edge $(u_i, v_j) \in E_1 (1 \le i, j \le N)$ if there is a packet of (i, j, d_1) in set A. The two bounds for this edge are $\alpha = 0$, and $\beta = p_{i,j}^{d_1}$, meaning that a legal flow $f(u_i, v_j)$ (an integer) on this edge must be between the two bounds and $f(u_i, v_j)$ packets will be dropped from set $S_{i,j}^{d_1}$. Fig. 4(a) shows the graph for the set A of Fig. 1, where, for clarity, the label on an edge is not shown if it is (0, 1).

Step 3 Construct a bipartite graph $G_2(U_2, V_2, E_2)$ for set B in the same way as for set A. Fig. 4(b) shows the graph for the set B of Fig. 1. For this graph, $U_2 = \{w_1, w_2, \ldots, w_N\}$ and $V_2 = \{z_1, z_2, \ldots, z_N\}$. The label for edge (w_i, z_j) is $(0, p_{i,j}^{d_2})$.

Step 4 Union the two graphs constructed in Step 2 and Step 3 into a single bipartite graph. Then add N vertices $\{x_1, x_2, \ldots, x_N\}$ to the left side of the graph and another N vertices $\{y_1, y_2, \ldots, y_N\}$ to the right side of the graph. Fig. 5 shows the construction obtained from the two graphs of Fig. 4. Moreover, two edges (x_i, u_i) and (x_i, w_i) are added for each $x_i (1 \le i \le N)$ and two edges (v_j, y_j) and (z_j, y_j) are added for each $y_j (1 \le i, j \le N)$.

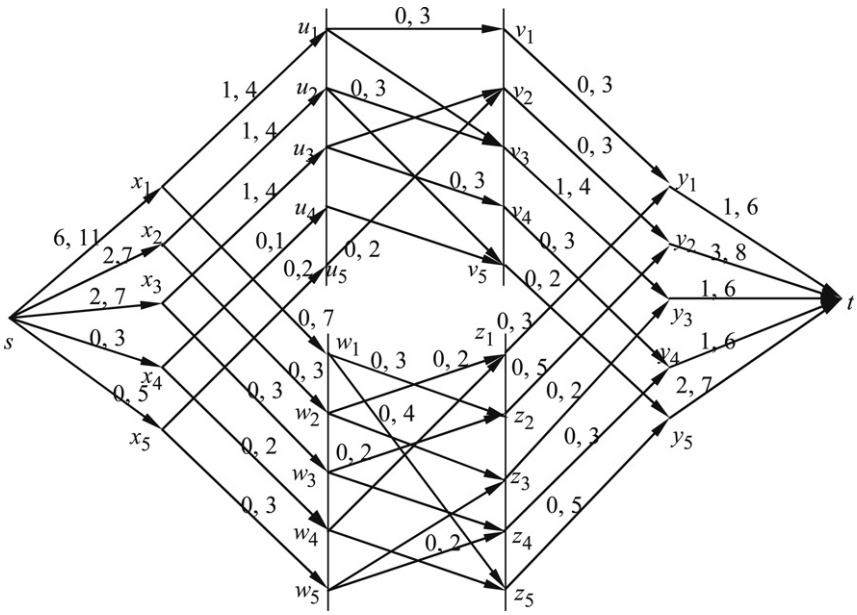


Fig. 6. The final network graph $G(V, E)$.

The two bounds of edge (x_i, u_i) are $\alpha = \max(0, a_i^{d_1})$, $\beta = p_i^{d_1}$, meaning that we must drop at least $a_i^{d_1}$ class A packets from input port i such that the remaining set can satisfy condition (3). β is the number of class A packets at input port i before dropping. The label of edge (x_i, w_i) is $(0, p_i^{d_2})$, meaning that we may need to drop some class B packets from input port i and there are a total of $p_i^{d_2}$ packets to choose from.

The label of edge (v_j, y_j) is $\alpha = \max(0, b_j^{d_1})$, $\beta = q_j^{d_1}$, meaning that we must drop at least $b_j^{d_1}$ class A packets that go to output port j but cannot be more than $q_j^{d_1}$. The label of edge (z_j, y_j) is $(0, q_j^{d_2})$, meaning that we may need to drop some class B packets that go to output port j and there are a total of $q_j^{d_2}$ packets to choose from.

Let us denote the graph constructed in this step by $G'(V', E')$.

Step 5 To finish the construction of the network graph $G(V, E)$, we add a source vertex s to the left side of graph $G'(V', E')$ and a sink vertex t to the right side of $G'(V', E')$. There is an edge from s to each x_i and an edge from each y_j to $t (1 \le i, j \le N)$. Fig. 6 shows the final network graph for the example of Fig. 1. The two bounds on edge (s, x_i) are $\alpha = \max[0, (p_i^{d_1} + p_i^{d_2}) - (d_2 + 1)] = \max(0, a_i^{d_1} + a_i^{d_2})$, $\beta = p_i^{d_1} + p_i^{d_2}$, meaning that we must drop at least a total of $a_i^{d_1} + a_i^{d_2}$ packets from input port i to satisfy condition (4). β is the total number of packets at input port i before dropping. Similarly, the two bounds on edge (y_j, t) are $\alpha = \max[0, (q_j^{d_1} + q_j^{d_2}) - (d_2 + 1)] = \max(0, b_j^{d_1} + b_j^{d_2})$, $\beta = q_j^{d_1} + q_j^{d_2}$.

Step 6 End.

From the explanations of each step in Procedure Network-Graph, it is easy to see that if we can find a legal flow for the network graph $G(V, E)$, then after dropping the packets according to the flow on each edge of the bipartite graph (in the middle of $G(V, E)$), the remaining packet set satisfies conditions (3) and (4). Conversely, if we can drop some packets to make the remaining set to satisfy conditions (3) and (4), then we can construct a corresponding legal flow from these dropped packets. Therefore, the problem becomes how to find a legal flow that is minimum. Obviously, a legal flow exists. For example, setting a flow on any edge to its upper bound is a legal flow, which corresponds to deleting all packets of A and B. By doing so, the remaining set becomes empty which satisfies conditions (3) and (4) obviously. What we need is the minimum legal integral flow which corresponds to the minimum dropping. This minimum legal flow can be found by an existing algorithm given by [17]. Moreover, the complexity of this algorithm remains the same as that for computing a maximum flow for a regular network, which is at most $O(N^3)$ [17,18]. We omit the details.

5. Conclusions

In this paper, we have proved that, if the traffic of both classes is overloaded, that is, both conditions (3) and (4) are not satisfied, then finding an optimal schedule for an input-queued switch is an NP-complete problem. To cope with the NP-hardness, this paper proposes an optimal preprocessing algorithm that shapes the traffic into a non-overloaded traffic by dropping a minimum number of packets. This result provides theoretical and practical guidelines to improve on the current

FIPS algorithm, a non-EDF based scheduling algorithm. As a future work, we will investigate the NP-completeness for the case if either condition (3) or (4) is satisfied but the other is not.

Acknowledgments

The work done by Xiaojun Shen was partially supported by the UMRB grant K9949, a contribution made by FutureNet Technology Inc., and K.C.Wong Education Foundation, Hong Kong. The work done by Jianyu Lou was partially supported by the UMRB grant K9949 and Dr. Khosrow Sohraby's personal research funding. The work done by Weifa Liang is fully funded by a research grant No: DP0449431 by Australian Research Council under its Discovery Schemes. The work by Junzhou Luo is supported by National Natural Science Foundation of China under Grant No. 90604004, Jiangsu Provincial Natural Science Foundation in China Grant No. BK2007708, and Jiangsu Provincial Key Laboratory of Network and Information Security Grant No. BM2003201.

References

[1] Y. Lee, J. Lou, J. Luo, X. Shen, An efficient packet scheduling algorithm with deadline guarantees for input-queued switches, IEEE/ACM Transactions on Networking 15 (1) (2007) 212–225.
[2] M. Karol, M. Hluchyj, S. Morgan, Input versus output queuing on a space-division packet switch, IEEE Transactions on Communications 35 (12) (1987) 1347–1356.
[3] N. McKeown, V. Anantharam, J. Warland, Achieving 100% throughput in an input-queued switch, in: Proc. of IEEE INFOCOM'96, pp. 296–302.
[4] A. Mekkittikul, N. McKeown, A practical scheduling algorithm to achieve 100% throughput in input-queued switches, in: Proc. of IEEE INFOCOM'98, pp. 792–799.
[5] A. Kam, K. Siu, Linear-complexity algorithms for QoS support in input-queued switches with no speedup, IEEE Journal on Selected Areas in Communications 17 (6) (1999) 1040–1056.
[6] N. Mckeown, The iSLIP scheduling algorithm for input-queued switches, IEEE/ACM Transactions on Networking 7 (2) (1999) 188–201.
[7] D. Stiliadis, A. Varma, Providing bandwidth guarantees in an input-buffered crossbar switch, in: Proc. of INFOCOM'95, pp. 960–968.
[8] A. Hung, G. Kesidis, N. Mckeown, ATM input-buffered switches with guaranteed rate property, in: Proc. of IEEE ISCC'98, July 1998, pp. 331–335.
[9] T. Lee, C. Lam, Path switching — A quasi-static routing scheme for large scale ATM packet switches, IEEE Journal on Selected Areas in Communications 15 (1997) 914–924.
[10] S. Li, N. Ansari, Input-queued switching with QoS guarantees, in: Proc. of IEEE INFOCOM'99, pp. 1152– 1159.
[11] C. Chang, W Chen, H. Huang, Birkhoff-von Neumann input buffered crossbar switches, in: Proc. of IEEE INFOCOM'2000, pp. 1614–1623.
[12] M.A. Bonuccelli, M.C. Clò, Scheduling of real-time messages in optical broadcast-and-select networks, IEEE/ACM Transactions on Networking 9 (5) (2001) 541–552.
[13] I.R. Philp, J.W. Liu, A switching problem for real time periodic messages, Available: Http://www-rtsl.cs.uiuc.edu/papers.
[14] V. Tabatabaee, L. Georgiadis, L. Tassiulas, QoS provisioning and tracking fluid policies in input queuing switches, IEEE/ACM Transactions on Networking 9 (5) (2001) 605–617.
[15] M.A. Bonuccelli, I. Gopal, C.K. Wong, Incremental time-slot assignment in SS/TDMA satellite systems, IEEE Transactions on Communications 39 (7) (1991) 1147–1156.
[16] Z.-L. Zhang, Z. Duan, Y.T. Hou, Virtual time reference system: A unifying scheduling framework for scalable support of guaranteed services, IEEE Journal on Selected Areas In Communications 18 (12) (2000) 2684–2695.
[17] Shimon Even, Graph Algorithms, Computer Science Press, 1979.
[18] T.H. Cormen, C.E. Leiserson, R.L. Rivest, Clifford Stein, Introduction to Algorithms, second ed., McGraw-Hill, 2001.