Buffer Management for Colored Packets with Deadlines[*]

Yossi Azar[†]　　　Uriel Feige[‡]　　　Iftah Gamzu[§]　　　Thomas Moscibroda[¶]

Prasad Raghavendra[‖]

Abstract

We consider buffer management of unit packets with deadlines for a multi-port device with reconfiguration overhead. The goal is to maximize the throughput of the device, i.e., the number of packets delivered by their deadline. For a single port or with free reconfiguration, the problem reduces to the well-known packets scheduling problem, where the celebrated earliest-deadline-first (EDF) strategy is optimal 1-competitive. However, EDF is not 1-competitive when there is a reconfiguration overhead. We design an online algorithm that achieves a competitive ratio of $1 - o(1)$ when the ratio between the minimum laxity of the packets and the number of ports tends to infinity. This is one of the rare cases where one can design an almost 1-competitive algorithm. One ingredient of our analysis, which may be interesting on its own right, is a perturbation theorem on EDF for the classical packets scheduling problem. Specifically, we show that a small perturbation in the release and deadline times cannot significantly degrade the optimal throughput. This implies that EDF is robust in the sense that its throughput is close to the optimum even when the deadlines are not precisely known.

1 Introduction

One of the main criticisms against competitive analysis is that it is unsuitable for studying real-life scenarios. Suppose one develops an online algorithm with relatively small competitive ratio, say, a competitive ratio of $1/2$. It turns out that this 50% performance loss is unacceptable by many practical models. Accordingly, when designing online algorithms for such models, the goal should be to attain a competitive ratio of nearly 1. Unfortunately, there are only few cases in which one can devise algorithms that achieve almost the same performance as an optimal offline algorithm. One well-known problem, which admits a 1-competitive online algorithm, is *packets scheduling*. This problem concerns with scheduling issues arising in context of network devices such as switches and routers. In this problem, there is an incoming stream of packets with deadlines. At any point in time, a pending packet may be transmitted, where the transmission of a packet takes unit time. The goal is to maximize the number of packets transmitted prior to their deadline time. The celebrated *earliest-deadline-first* (EDF) strategy is known to attain optimal throughput for instances of this problem. However, the above-mentioned scenario overlooks many factors which are of essence in practice. In particular, it fails to model scheduling issues arising when the network device has multiple outgoing ports, and there is a reconfiguration overhead for switching between these ports.

This paper deals with overhead issues arising in context of multi-port network devices. The general scenario is similar to that of packets scheduling. There is an incoming stream of time-dependent packets. Each of the packets has a designated outgoing port. The network device maintains an active outgoing port on which pending packets may be transmitted, and there is a reconfiguration overhead of unit time taking the network device to switch the active outgoing port. The goal is to maximize the throughput of transmitted packets, that is, to transmit a maximum number of packets along their designated output port prior to their deadline time. We present an online algorithm that has a competitive ratio of nearly 1, for a wide range of practical parameters. Specifically, we demonstrate that if the ratio between the minimum laxity of the packets and the number of outgoing ports tends to infinity then our algorithm achieves a competitive ratio of $1 - o(1)$.

We formally model the above-mentioned scenario of *buffer management for colored packets with deadlines* as follows. There is an online sequence of packets σ. A packet $i \in \sigma$ is characterized by a triplet (r_i, d_i, c_i), where $r_i \in \mathbb{N}_+$ and $d_i \in \mathbb{N}_+$ are the respective release time and deadline time of the packet, and c_i is the color or designated output port of the packet. A packet becomes known to the system at its release time. The objective is to schedule the packets in a way that maximizes the number of transmitted packets, while maintaining the following two properties. The first property is that every scheduled packet i is transmitted during the time frame $[r_i, d_i]$, and the second property is that there is a time-slot dedicated to a *color transition* between the transmission of two successive packets with different colors.

1.1 Our results

The main result of this paper is a deterministic online algorithm that attains a competitive ratio of $1 - o(1)$ when the ratio between the minimum laxity of the packets and the number of outgoing ports tends to infinity, where the *laxity* of packet i is defined as $d_i - r_i$. Our algorithm strikes a balance between two natural approaches: a strategy that transmits packets whose deadline approaches, and a strategy which makes an effort to minimize the color transitions. In addition, we prove that the restrictions imposed by this setting are essential to almost match the optimal offline throughput. Specifically, we demonstrate that if the number of different colors and the minimum laxity is of the same order then there is some universal constant $\alpha < 1$ that bounds the achievable competitive ratio.

This bound also applies for randomized online algorithms. Finally, we demonstrate that the offline version of the problem is NP-hard. This result holds even when all laxities are equal.

One ingredient of our analysis, which may be interesting on its own right, is a perturbation theorem for the fundamental problem of packets scheduling. Interestingly, we prove that a small perturbation in the release and deadline times of packets cannot induce a major degradation in the optimal throughput. More precisely, we show that the optimal throughput does not degrade by more than a multiplicative factor of $1 - \delta$, where δ is the perturbation time divided by the minimum laxity.

1.2 Related work

In the following, we provide a brief overview of two extensively-studied buffer management models: the bounded delay model, and the FIFO-queue model. Due to the ever-growing line of work in this context, it is beyond the scope of this writing to do justice and present an exhaustive survey of all previous work. We refer the reader to the cited papers and the references therein for a more comprehensive review of the literature.

In the bounded delay model [21], there is an incoming stream of packets, each of which has an intrinsic value, and the goal is to maximize the value of the packets transmitted prior to their expiration time. It is well-known that the EDF strategy is optimal when the values of all packets are identical. Turning to the arbitrary values case, the best known deterministic online algorithm has a competitive ratio of about 0.547 [15] (see also [28]), while it is known that no deterministic online algorithm can achieve a competitive ratio better than $1/\phi \approx 0.618$ [20, 9, 3]. Focusing on the randomized setting, the best online algorithm attains a ratio of $1 - 1/e \approx 0.632$ [6, 8], while it is known that no online algorithm can attain a ratio better than 0.8 [9]. Several additional papers addressing the bounded delay model or its variants are [27, 10, 7]. Kesselman et al. [21] also initiated the study of the FIFO-queue model. In this model, there is also an incoming stream of heterogeneous packets. Nevertheless, packets have no deadlines, they are placed in a limited-size FIFO-queue, and have to be transmitted according to their arrival order. The main difficulty in this model is to decide which packets should be accepted to the transmission queue. Some of the many papers studying this model and its variants are [2, 4, 1, 22, 32, 14, 16].

A concurrent line of work, initiated by Räcke, Sohler and Westermann [30], studied the sorting buffer problem. An instance of this problem consists of a server, equipped with a finite-capacity buffer, and an online sequence of requests, each of which corresponds to a point in a metric space. The goal is to serve all requests in a way that minimizes the total distance traveled by the server. Note that the server can store requests in the buffer and then serve them in an out-of-order fashion. One may reinterpret this problem in the context of multi-port devices as follows: there is an incoming stream of packets that needs to be served, and the goal is to design a scheduling policy which uses a reordering buffer and minimizes the overall reconfiguration costs. A special case of interest is when the metric space is uniform, or equivalently, when reconfiguration costs are identical. Several papers attending to the sorting buffer problem are [13, 25, 12, 24, 23, 5, 17, 11].

1.3 Application to DRAM memory scheduling

Our original impetus for studying the above problem is an application to memory request scheduling in DRAM systems. Modern DRAM chips are organized into different *banks*, so that memory requests destined for different banks can be served in parallel. Each DRAM bank has a two-dimensional structure, consisting of rows and columns (see, for instance, [31, 29]). Consecutive memory addresses are located in consecutive columns in the same row. The size of a row varies, but it is usually between

1-32Kbytes ($\sim$ 32-1024 L2 cache blocks) in commodity DRAMs.

Each bank has a *row-buffer*, and data can only be read from this buffer. At any given time, the row-buffer contains a single row. Depending on the access pattern to a bank, a DRAM request can either be a *row-hit*, in case the request is to a row that is currently in the row-buffer, or a *row-conflict*, if the request is to a row different than the one that is currently in the row-buffer. In case of a row-hit, the request can simply be executed to the row-buffer. In contrast, in case of a row-conflict, the request may only be executed after the current row in the row-buffer is written back to the memory and the requested row is loaded into the row-buffer. On commodity DRAM, row-hit and row-conflict accesses typically consume 40-50ns and 80-100ns, respectively.[1]

Today's microprocessors employ hardware prefetchers to hide long DRAM access latencies (see, e.g., [26]). If prefetch requests are accurate and fetch data early enough, prefetching can improve performance. However, a prefetch request is helpful only if it is served (by the DRAM memory) sufficiently fast so that the processor need not stall. Each prefetch request is therefore effectively associated with a pre-defined deadline after its release; no benefit results from serving the request after the deadline. The combination of row-buffer switching costs and the deadlines for DRAM requests maps to our problem definition. Specifically, we seek to maximize the number of successfully scheduled prefetch requests. Note that we focus on the one-bank case since every DRAM bank may be scheduled independently.

1.4   Structure of the paper

In Section 2, we consider the classical packets scheduling problem, and show that a small perturbation in the release and deadline times cannot significantly decrease the optimal throughput of the sequence. In Section 3, we describe the online $(1-o(1))$-competitive algorithm for our problem and its analysis. In Section 4, we demonstrate that if the number of colors is not small compared with the minimum laxity of the packets then there is a constant bound on the attainable competitive ratio. Finally, details omitted from the main part of the paper, including the NP-hardness result for the offline version, appear in the Appendix.

2   Perturbation Theorem for the Packets Scheduling Problem

In this section, we consider the classical packet scheduling problem in which packets have no colors (or equivalently, have a single color). Hence, a packet is represented by a release-deadline pair (r, d). Let $\text{OPT}(\sigma)$ be the throughput of the optimal schedule with respect to σ, that is, the schedule that is produced by EDF, and let $L = \min_{i \in \sigma}\{d_i - r_i\}$ denote the minimum laxity of the packets. We show that a small perturbation of the release and deadline times of the packets in the sequence cannot significantly degrade the throughput when L is large. This can also be viewed as a perturbation theorem on EDF. Specifically, if one schedules the packets according to a perturb EDF, the throughput is not significantly reduced compared to the throughput of a schedule generated with an exact EDF.

Definition 2.1. An input sequence $\hat{\sigma}$ is a λ-*perturbation* of σ if $\hat{\sigma}$ consists of all packets of σ, and each packet $(\hat{r}, \hat{d}) \in \hat{\sigma}$ corresponding to packet $(r, d) \in \sigma$ satisfies $\hat{r} - r \leq \lambda$ and $d - \hat{d} \leq \lambda$.

Theorem 2.2. *Suppose $\hat{\sigma}$ is a λ-perturbation of σ then* $\text{OPT}(\hat{\sigma}) \geq (1 - 2\lambda/L) \cdot \text{OPT}(\sigma)$.

---

[1]Notice that due to the existence of the row-buffer, modern DRAMs are not truly random access (i.e., having equal access time to all locations in the memory array).

Proof. In what follows, we argue about the case that $\hat{\sigma}$ is a 1-perturbation of σ. We demonstrate that an optimal EDF schedule for σ having throughput τ can be adjusted to be feasible for $\hat{\sigma}$, while maintaining a throughput of at least $(L-2)/L \cdot \tau$. This implies that the optimal schedule for $\hat{\sigma}$ has at least as large throughput. The lemma follows by viewing the λ-perturbation as a series of 1-perturbations, and employing the above-mentioned argument λ times. Particularly, one should pay attention that after 1-perturbation of some input sequence, its minimum laxity may decrease by 2; one resulting from the perturbation in the deadline times, and one resulting from the perturbation in the release times. Therefore, after λ applications of the above-mentioned argument, the resulting schedule has a throughput of at least

$$\left(\frac{L-2\lambda}{L-2\lambda+2} \cdots \left(\frac{L-4}{L-2} \left(\frac{L-2}{L} \cdot \tau \right) \right) \cdots \right) = \frac{L-2\lambda}{L} \cdot \tau \ .$$

Let ρ be the optimal EDF schedule of the input sequence σ, having a throughput of τ. We partition ρ into *super-blocks*, each of which is a maximum consecutive sequence with no idle time. Then, we apply the subsequent arguments separately to each of these super-blocks. We begin by demonstrating that each super-block can be adjusted to satisfy the perturbed deadline times. For this purpose, consider the partition of a super-block into consecutive blocks, each consisting of L successive time-slots. Let us focus on one of these blocks, and let $t_1 < \cdots < t_k$ be the collection of time-slots in which there are packets which are no longer feasible for transmission due to the perturbation of the deadline times. We adjust the schedule of this block by dropping the packet in time-slot t_1, and relocating each packet in time-slot t_{i+1} to time-slot t_i, for any $i \in [k-1]$. Figure 1 provides a schematic illustration of this modification process. One should notice that the resulting schedule is feasible with respect to the perturbed deadline times. This follows from the fact that each packet, which was infeasible for transmission, was either dropped or relocated to a strictly prior time-slot. Moreover, note that each relocated packet is still transmitted after its release time as its transmission time decreased by no more than $L-1$, whereas the minimum laxity of a packet is L. Focusing on the throughput, it is easy to see that no more than one packet was dropped in each partition block. In fact, one can reinforce the last observation, and to additionally claim that no packet was dropped from the first partition block. This claim holds since the first partition block spans the time frame $[1, L]$, but the deadline time of any packet is at least $L+1$. Consequently, no more than $\lfloor \tau/L \rfloor$ packets were dropped during this modification process, and hence, the throughput of the new schedule is at least $\tau' = (L-1)/L \cdot \tau$.

Now, we may assume that the optimal EDF schedule of the input sequence that incorporates the perturbation of the deadline times has a throughput of at least τ'. We turn to adjust this schedule to satisfy the perturbed release times. Essentially, this can be done in a similar way to that of the perturbed deadline times. In this case, the last packet in each partition block is dropped, and packets are relocated to a strictly later time-slot. It seems worthy to note that now, a packet may be dropped from the first partition block, but can be spared from the last partition block as the last infeasible packet may be relocated to the end of the schedule while maintaining deadline-feasibility. Correspondingly, we obtain that the throughput of the new schedule is at least $(L-2)/(L-1) \cdot \tau' = (L-2)/L \cdot \tau$. The $(L-2)/(L-1)$ term results by recalling that the initial schedule has a minimum laxity of $L-1$ (and not necessarily L). ∎

3 Online Scheduling for Colored Packets

In this section, we design a deterministic online algorithm that achieves a competitive ratio of $1-o(1)$ when the number of different packet colors is asymptotically small with respect to the minimum laxity
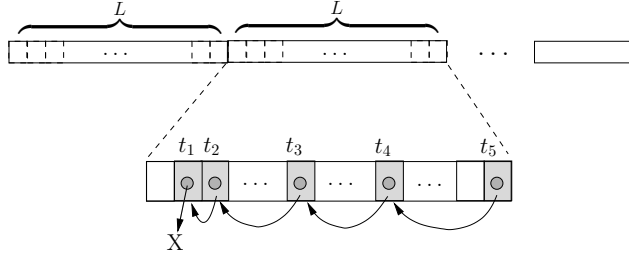
Figure 1: The modification of the schedule with respect to the perturbed deadline times.

of the packets. It is worthy to note that this setting is essential to design a $(1 - o(1))$-competitive algorithm, as established in Section 4. Let $C = |\{c_i : i \in \sigma\}|$ be the number of different packet colors, and recall that $L = \min_{i \in \sigma}\{d_i - r_i\}$ is the minimum laxity of the packets.

We begin by observing that several natural greedy algorithms fail to attain near-optimal throughput in the above-mentioned setting. In particular, if we schedule packets by providing priority to packets with lower deadlines (as done by EDF), or alternatively, if we give priority to packets with the same color as the color of the currently transmitted packet, then we cannot attain a competitive ratio better than $1/2$. These results hold even when $C = o(L)$. Further details appear in Appendix A.

3.1   The algorithm

In light of the insufficiency of the natural algorithms to achieve a competitive ratio of $1 - o(1)$, it seems essential for a good strategy to strike a balance between transmitting packets whose deadline approaches, and transmitting packets whose color is identical to the current active color. We present a deterministic online algorithm that realizes this insight. The algorithm attains a competitive ratio of $1 - 4\sqrt{C/L}$. In particular, this implies that it attains almost optimal throughput for instances having $C = o(L)$.

Algorithm *balanced-greedy* (BG), formally described in Figure 2, works in phases, each of which spans a time frame of $K = \sqrt{CL}$ time-slots. Every phase is logically built from a *collection step*, followed by a *scheduling step*. Specifically, during the collection step, the algorithm accumulates all the packets released in the phase, and during the scheduling step, the algorithm sets a transmission schedule for the K time-slots of the next phase.

Analysis. To establish the correctness of the algorithm, we need to demonstrate that the output schedule ρ is feasible. Specifically, one needs to prove that every scheduled packet i is transmitted during the time frame $[r_i, d_i]$, and that there is a color transition between the transmission of any two successive packets with different colors.

Lemma 3.1. *Algorithm BG generates a valid schedule.*

Proof. By the construction of the schedule, it follows that there is a color transition between the transmission of any two consecutive packets with different colors, and that every packet is transmitted after its arrival. Hence, we are left to argue that every packet is transmitted before its deadline. Consider some scheduled packet i. Notice that the deadlines modification during the collection step ensures that the deadline of each packet is aligned with the end time of some phase. Consequently, if it is feasible to transmit i during a specified time-slot of some phase then it is feasible to transmit it in any of the time-slots of that phase. This implies that the reordering applied to the K-length prefix of the EDF schedule during the scheduling step does not change its feasibility. ∎

5

> Let ρ be an empty output schedule.
>
> In each phase $\ell = 1, 2, \ldots$ do
>
> **The collection step:** Collect all packets released in time frame $(K(\ell - 1), K\ell]$. Modify the deadline of each collected packet (r, d, c) from d to $K \cdot \lfloor d/K \rfloor$.
>
> **The scheduling step:** Let S^ℓ be the collection of packets appearing in the K-length prefix of the current EDF schedule of pending packets (i.e., packets arrived at phases 1 to ℓ that were not already transmitted or dropped), and let $S_j^\ell \subseteq S^\ell$ denote the subset of packets having color c_j. Append the output schedule ρ with the K-length schedule ρ_ℓ, constructed as follows:
>
> - ρ_ℓ initially consists of all packets of S_1^ℓ scheduled consecutively, then all packets of S_2^ℓ scheduled consecutively, and so on.
>
> - ρ_ℓ is altered to have a color transition between each two successive color groups S_j^ℓ, S_{j+1}^ℓ: if the overall length of the schedule is less than K then a dedicated color transition time-slot is added after the last packet of S_j^ℓ and before the first packet of S_{j+1}^ℓ; otherwise, the last packet of S_j^ℓ is dropped and replaced by a color transition.
>
> - ρ_ℓ is affixed with idle time-slots, so its length will be exactly K.

Figure 2: Algorithm BG.

We now turn to analyze the performance guarantee of the algorithm. We begin by introducing some notation and terminology:

- Let $\rho(\sigma)$ denote the output schedule of algorithm BG with respect to input sequence σ. Note that the output sequence consists of two types of time-slots: *productive* time-slots in which packets are transmitted, and *non-productive* time-slots, which are either idle time-slots or color transition time-slots. Lastly, we call $\rho(\sigma)$ a *lazy* schedule if it consists of at least one idle time-slot besides the first K time-slots which are always idle.

- Let $\mathrm{ON}(\sigma)$ and $\mathrm{OPT}(\sigma)$ denote the throughput of the schedule generated by algorithm BG and the optimal schedule with respect to σ, respectively.

- Let $\mathrm{LEN}(\sigma)$ be the length of the output schedule generated with respect to σ, that is, $\rho(\sigma)$. Moreover, let $\mathrm{IDLE}(\sigma)$ and $\mathrm{TRANS}(\sigma)$ be the number of idle time-slots and color transition time-slots of $\rho(\sigma)$, respectively. Notice that $\mathrm{LEN}(\sigma) = \mathrm{ON}(\sigma) + \mathrm{IDLE}(\sigma) + \mathrm{TRANS}(\sigma)$.

We proceed by proving that if the length of the schedule generated by algorithm BG is sufficiently short then it has near-optimal throughput.

Lemma 3.2. *Algorithm BG achieves a competitive ratio of $1 - \sqrt{C/L}$, when the length of its generated schedule is less than L.*

Proof. Recall that the deadline time of any packet is no less than $L + 1$. Accordingly, we know that no packet was discarded due to its deadline. In particular, this implies that the sole reason that some packet was dropped was to capacitate a color transition. However, no more than C packets may be dropped in each phase, and if a packet is dropped then there must be at least $K - C$ packets transmitted during this phase. Consequently, the relative throughput of the algorithm in each phase is at least $(K - C)/K = 1 - \sqrt{C/L}$, where the equality holds since $K = \sqrt{CL}$. ∎

In the following, we establish that algorithm BG also has near-optimal throughput when the generated schedule is long. We begin with the following lemma which demonstrates that for purpose of analysis, it is sufficient to consider input sequences for which algorithm BG generates non-lazy output schedules.

Lemma 3.3. *It is sufficient to analyze the performance guarantee of algorithm BG with respect to input sequences for which it generates non-lazy schedules.*

Proof. Consider some input sequence σ for which the output schedule $\rho(\sigma)$ is lazy, and let us assume without loss of generality that an idle time-slot appears in the time frame corresponding to phase $\ell > 1$. Let $\langle \sigma_1, \sigma_2 \rangle$ be a partition of σ such that σ_1 consists of all packets released during the first $\ell - 1$ phases, and σ_2 consists of the remaining packets. The fact that there is an idle time-slot in the time frame corresponding to phase ℓ ensures that no packet of σ_1 is scheduled after that phase. In particular, this implies that $\rho(\sigma_1)$ is identical to the ℓ-phases prefix of $\rho(\sigma)$. Furthermore, one can easily validate that when one drops the first K idle time-slots of $\rho(\sigma_2)$ then the resulting schedule is identical to the suffix of $\rho(\sigma)$ that excludes the first ℓ phases. Thus, the performance guarantee of algorithm BG with respect to at least one of $\{\sigma_1, \sigma_2\}$ must bound the performance guarantee of the algorithm with respect to σ. It is clear that one can inductively apply the above-mentioned argument with respect to σ_1 or σ_2 in case $\rho(\sigma_1)$ or $\rho(\sigma_2)$ are lazy. Hence, there is an input sequence for which algorithm BG generates a non-lazy schedule that bounds the performance guarantee of the algorithm with respect to σ. ∎

We proceed by lower-bounding the throughput of the algorithm. The following lemma argues the throughput of the algorithm is nearly the length of the generated schedule.

Lemma 3.4. $\mathrm{ON}(\sigma) \geq \left(1 - 2\sqrt{C/L}\right) \cdot \mathrm{LEN}(\sigma)$.

Proof. In what follows, we establish two claims that may be used, in conjunction with the facts that $\mathrm{ON}(\sigma) = \mathrm{LEN}(\sigma) - \mathrm{IDLE}(\sigma) - \mathrm{TRANS}(\sigma)$ and $K = \sqrt{CL}$, to prove the lemma.

Claim I: $\mathrm{IDLE}(\sigma) \leq \mathrm{LEN}(\sigma) \cdot K/L$. By Lemma 3.3, we may assume that $\rho(\sigma)$ is non-lazy. Therefore, the only idle time-slots in $\rho(\sigma)$ are the first K time-slots, and hence, $\mathrm{IDLE}(\sigma) = K$. In addition, by Lemma 3.2, we may assume that $\mathrm{LEN}(\sigma) \geq L$, which implies that $K \leq K \cdot \mathrm{LEN}(\sigma)/L$.

Claim II: $\mathrm{TRANS}(\sigma) \leq \mathrm{LEN}(\sigma) \cdot C/K$. Notice that there are at most $\lceil \mathrm{LEN}(\sigma)/K \rceil - 1$ phases in which algorithm BG transmits packets. Note that the -1 term results from the fact that the algorithm does not transmit any packet during the first phase. Since there are no more than C color transitions in each phase, we obtain that $\mathrm{TRANS}(\sigma) \leq C \cdot (\lceil \mathrm{LEN}(\sigma)/K \rceil - 1) \leq C \cdot \mathrm{LEN}(\sigma)/K$. ∎

We now turn to prove that the length of the schedule is almost equivalent to the throughput of the optimal schedule. We first define two input sequence σ' and $\tilde{\sigma}$, which are modifications of σ. The input sequence σ' consists of all packets in σ, but alters the color of packets to some fixed color c' (or equivalently, no color at all). Specifically, each packet $(r, d, c) \in \sigma$ defines a packet $(r, d, c') \in \sigma'$. The input sequence $\tilde{\sigma}$ consists of all packets in σ such that a packet $(r, d, c) \in \sigma$ gives rise to a packet $(K \cdot \lceil r/K \rceil, K \cdot \lfloor d/K \rfloor, c') \in \tilde{\sigma}$, where c' is a some fixed color as before. Essentially, all packets in $\tilde{\sigma}$ have the same color, and the release and deadline times of each packet in $\tilde{\sigma}$ are aligned with start/end time of the corresponding phase in a way that the span of each packet is fully contained in the span of that packet according to σ. Remark that the *span* of a packet (r, d, c) is the time frame $[r, d]$.

Lemma 3.5. $\mathrm{LEN}(\sigma) = \mathrm{OPT}(\tilde{\sigma})$.

Proof. Recall that a K-length prefix of the EDF schedule of pending packets is considered at the end of any phase, and utilized to construct the schedule of the subsequent phase. Let us focus on the ordered concatenation of these K-length EDF sub-schedules, and argue that this concatenated schedule is identical to the schedule generated by EDF with respect to $\tilde{\sigma}$, excluding its initial K idle time-slots. For this purpose, we consider the algorithm's EDF sub-procedure, and prove that it experiences the packets as if their release and deadline times were determined by $\tilde{\sigma}$. Observe that each packet may be scheduled by the algorithm in a phase later than its arrival phase. Taking the EDF sub-procedure point of view, this implicitly corresponds to a modification of the release time of each packet to the start time of the next phase, similarly to the alteration of the release time realized in $\tilde{\sigma}$. Turning to the deadlines, the algorithm explicitly modifies them in an identical way to that done in $\tilde{\sigma}$.

Now, notice that the length of the schedule generated by the algorithm, excluding the initial K idle time-slots, is equal to the overall lengths of all EDF sub-schedules considered by the algorithm. This implies, in conjunction with the previous-mentioned argument, that the length of the schedule generated by the algorithm is equal to the length of the schedule generated by EDF with respect to $\tilde{\sigma}$. The lemma follows by recalling that EDF generates optimal schedules for instances in which all packets have the same color. ∎

A crucial step in the analysis is to notice that $\tilde{\sigma}$ is a K-perturbation of σ', and the colors of all packets are identical. Hence, by using Theorem 2.2 we obtain the following corollary.

Corollary 3.6. $\mathrm{OPT}(\tilde{\sigma}) \geq \left(1 - 2\sqrt{C/L}\right) \cdot \mathrm{OPT}(\sigma')$.

We are now ready to prove the main theorem of this section.

Theorem 3.7. *Algorithm BG attains a competitive ratio of at least* $1 - 4\sqrt{C/L}$.

Proof. Using the previously stated results, we obtain that

$$
\begin{aligned}
\mathrm{ON}(\sigma) &\geq \left(1 - 2\sqrt{C/L}\right) \cdot \mathrm{LEN}(\sigma) \\
&= \left(1 - 2\sqrt{C/L}\right) \cdot \mathrm{OPT}(\tilde{\sigma}) \\
&\geq \left(1 - 2\sqrt{C/L}\right)^2 \cdot \mathrm{OPT}(\sigma') \\
&\geq \left(1 - 4\sqrt{C/L}\right) \cdot \mathrm{OPT}(\sigma) \ .
\end{aligned}
$$

The first inequality results from Lemma 3.4. The equality follows from Lemma 3.5. The second inequality holds by Corollary 3.6. Finally, the last inequality follows as σ' is alike σ, but all packets have the same color. This implies that any schedule feasible for σ is also feasible for σ', and thus, $\mathrm{OPT}(\sigma') \geq \mathrm{OPT}(\sigma)$. ∎

4 Lower Bounds for Colored Packets Scheduling

In this section, we demonstrate that the restrictions imposed by the setting described in Section 3 are essential to almost match the optimal offline throughput. Specifically, we prove that there is no hope that a deterministic or randomized online algorithm can generate a schedule whose throughput is close to the optimal offline outcome, when the number of different colors is large with respect to the minimum laxity. We begin by presenting an simple lower bound of $1/2$ on the competitive ratio

of any deterministic online algorithm using an input sequence having $C = 2$ and $L = 0$. However, one may be left wondering if this lower bound is an artifact of the degenerated minimum laxity, and hence, it is possible to achieve better competitive ratio for larger values of L. Moreover, one may think that randomization can help. Therefore, we later prove a constant lower bound on the achievable competitive ratio of any deterministic or randomized online algorithm for any value of L, as long as $C = \Omega(L)$. Hence, our requirement that $C = o(L)$ is essential.

Theorem 4.1. *No deterministic online algorithm can achieve a competitive ratio better than $1/2$.*

Proof. Consider some online algorithm, and let $(1, 1, c_1)$ and $(1, 1, c_2)$ be two packets released at the first time-step with immediate deadline and different colors. Let us assume without loss of generality that the algorithm transmits the packet with color c_1 in the first time-slot. Subsequently, the packet $(2, 2, c_2)$ arrives. Clearly, the algorithm cannot transmit it since a color transition is required. This implies that the throughput of the schedule generated by the algorithm is exactly 1, whereas an the optimal schedule could transmit the two packets of color c_2. Notice that in case that the algorithm does not schedule any packet in the first time-slot, similar arguments apply. The lemma follows by noticing that we may use this building block repeatedly. Specifically, we use this block in time-slots 1 and 2, then leave time-slot 3 free for a possible color transition, use this block again in time-slots 4 and 5, and so on. ∎

Theorem 4.2. *No deterministic or randomized online algorithm can attain a competitive ratio better than some constant $\alpha < 1$ for any L, when $C = \Omega(L)$. Specifically, $\alpha \leq 1 - C/(8L)$ when $C \leq L/2$, and $\alpha \leq 15/16$, otherwise.*

Proof. We begin by constructing a packets sequence σ. We assume without loss of generality that $C \leq L/2$ since otherwise, one may use packets with only $C = L/2$ colors, and obtain a lower bound of at least $1 - C/(8L) = 15/16$. The sequence σ consists of C packets with distinct colors $c_1, \ldots, c_C$ whose release and deadline times are 1 and $L + C$, respectively. Furthermore, σ also consists of $L - C$ packets which are released at time-step C and have a laxity of L, that is, their deadline time is $L + C$. These packets are partitioned into $C/2$ groups, each having $2(L - C)/C$ packets, where all packets in each group have the same color, and groups have distinct *random* colors from $\{c_1, \ldots, c_C\}$.

Consider a random algorithm. One may assume without loss of generality that this algorithm transmits some $C/2$ packets during the first C time-slots. This assumption is justified since it is always better to schedule some packets than leave idle time-slots. Notice that in expectation $C/4$ of the packets that were not transmitted during that time frame have the same color as packets released at time-step C. Note that the best thing the algorithm can do is to pair each of these packets with the group of packets having the same color, and transmit them consecutively. This can be done during a time frame of $C/4 \cdot (1 + 1 + 2(L - C)/C) = L/2$ time-slots (e.g., between time-slots $C + 1$ and $C + L/2$), where it takes $C/4 \cdot (1 + 2(L - C)/C)$ time-slots to transmit all these packets and additional $C/4$ time-slots for color changes. In addition, the algorithm may transmit the remaining $C/4$ groups of packets during a time frame of $C/4 \cdot (1 + 2(L - C)/C) = L/2 - C/4$ time-slots (e.g., between time-slots $C + L/2 + 1$ and $3C/4 + L$), and an additional $(C/4)/2 = C/8$ distinct-colored packets up to time $L + C$. Consequently, we obtain that the expected number of packets that the algorithm transmitted is $L - C/8$.

Now, the optimal schedule can transmit all L packets. This schedule transmits the $C/2$ packets released at time-step 1, which do not have a color matching group released at time-step C, during the first C time-slots. On the following L time-slots, it transmits all the remaining packets by pairing each packet released at time-slot 1 with the group of packets having the same color, and transmitting them consecutively. Accordingly, we attain a lower bound of $\alpha \leq (L - C/8)/L = 1 - C/(8L)$. ∎

9

5 Concluding Remarks

Non-unit reconfiguration overhead. Recall that our model assumes that both the transmission of a packet and a color transition takes identical amount of time. One natural question to ask is what happens in case that the transmission of a packet is still unit, but a color transition takes W units of time. It turns out that algorithm BG can be easily refined to deal with this scenario while achieving a competitive ratio of $1 - O\big(\sqrt{CW/L}\big)$. This implies that it obtains almost optimal throughput for instances having $CW = o(L)$. The key modifications required in the algorithm are to set $K = \sqrt{CWL}$, and to update the alteration of each sub-schedule to have a color transition of length W between any two successive color groups. Further details are deferred to the full version of this paper.

References

[1] W. Aiello, Y. Mansour, S. Rajagopolan, and A. Rosén. Competitive queue policies for differentiated services. *J. Algorithms*, 55(2):113–141, 2005.

[2] N. Andelman and Y. Mansour. Competitive management of non-preemptive queues with multiple values. In *Proceedings 17th International Conference on Distributed Computing*, pages 166–180, 2003.

[3] N. Andelman, Y. Mansour, and A. Zhu. Competitive queueing policies for qos switches. In *Proceedings 14th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 761–770, 2003.

[4] N. Bansal, L. Fleischer, T. Kimbrel, M. Mahdian, B. Schieber, and M. Sviridenko. Further improvements in competitive guarantees for qos buffering. In *Proceedings 31st International Colloquium on Automata, Languages and Programming*, pages 196–207, 2004.

[5] R. Bar-Yehuda and J. Laserson. Exploiting locality: Approximating sorting buffers. *Journal of Discrete Algorithms*, 5(4):729–738, 2007.

[6] Y. Bartal, F. Y. L. Chin, M. Chrobak, S. P. Y. Fung, W. Jawor, R. Lavi, J. Sgall, and T. Tichý. Online competitive algorithms for maximizing weighted throughput of unit jobs. In *Proceedings 21st Annual Symposium on Theoretical Aspects of Computer Science*, pages 187–198, 2004.

[7] M. Bienkowski, M. Chrobak, C. Dürr, M. Hurand, A. Jez, L. Jez, and G. Stachowiak. Collecting weighted items from a dynamic queue. In *Proceedings 20th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1126–1135, 2009.

[8] F. Y. L. Chin, M. Chrobak, S. P. Y. Fung, W. Jawor, J. Sgall, and T. Tichý. Online competitive algorithms for maximizing weighted throughput of unit jobs. *J. Discrete Algorithms*, 4(2):255–276, 2006.

[9] F. Y. L. Chin and S. P. Y. Fung. Online scheduling with partial job values: Does timesharing or randomization help? *Algorithmica*, 37(3):149–164, 2003.

[10] M. Chrobak, W. Jawor, J. Sgall, and T. Tichý. Improved online algorithms for buffer management in qos switches. *ACM Transactions on Algorithms*, 3(4), 2007.

[11] M. Englert, H. Räcke, and M. Westermann. Reordering buffers for general metric spaces. In *Proceedings 39th Annual ACM Symposium on Theory of Computing*, pages 556–564, 2007.

[12] M. Englert, H. Röglin, and M. Westermann. Evaluation of online strategies for reordering buffers. In *Proceedings 5th International Workshop on Experimental Algorithms*, pages 183–194, 2006.

[13] M. Englert and M. Westermann. Reordering buffer management for non-uniform cost models. In *Proceedings 32nd International Colloquium on Automata, Languages and Programming*, pages 627–638, 2005.

[14] M. Englert and M. Westermann. Lower and upper bounds on fifo buffer management in qos switches. In *Proceedings 14th Annual European Symposium on Algorithms*, pages 352–363, 2006.

[15] M. Englert and M. Westermann. Considering suppressed packets improves buffer management in qos switches. In *Proceedings 18th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 209–218, 2007.

[16] A. Fiat, Y. Mansour, and U. Nadav. Competitive queue management for latency sensitive packets. In *Proceedings 19th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 228–237, 2008.

[17] I. Gamzu and D. Segev. Improved online algorithms for the sorting buffer problem. In *Proceedings 24th Annual Symposium on Theoretical Aspects of Computer Science*, pages 658–669, 2007.

[18] M. R. Garey and D. S. Johnson. Complexity results for multiprocessor scheduling under resource constraints. *SIAM J. Comput.*, 4(4):397–411, 1975.

[19] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

[20] B. Hajek. On the competitiveness of online scheduling of unit-length packets with hard deadlines in slotted time. In *Proceedings Conference on Information Sciences and Systems*, pages 434–438, 2001.

[21] A. Kesselman, Z. Lotker, Y. Mansour, B. Patt-Shamir, B. Schieber, and M. Sviridenko. Buffer overflow management in qos switches. *SIAM J. Comput.*, 33(3):563–583, 2004.

[22] A. Kesselman, Y. Mansour, and R. van Stee. Improved competitive guarantees for qos buffering. *Algorithmica*, 43(1-2):63–80, 2005.

[23] R. Khandekar and V. Pandit. Offline sorting buffers on line. In *Proceedings 17th International Symposium on Algorithms and Computation*, pages 81–89, 2006.

[24] R. Khandekar and V. Pandit. Online sorting buffers on line. In *Proceedings 23rd Annual Symposium on Theoretical Aspects of Computer Science*, pages 584–595, 2006.

[25] J. S. Kohrt and K. Pruhs. A constant approximation algorithm for sorting buffers. In *Proceedings 6th Latin American Symposium on Theoretical Informatics*, pages 193–202, 2004.

[26] C. J. Lee, O. Mutlu, V. Narasiman, and Y. N. Patt. Prefetch-aware DRAM controllers. In *Proceedings 41st Annual IEEE/ACM International Symposium on Microarchitecture*, pages 200–209, 2008.

[27] F. Li, J. Sethuraman, and C. Stein. An optimal online algorithm for packet scheduling with agreeable deadlines. In *Proceedings 16th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 801–802, 2005.

[28] F. Li, J. Sethuraman, and C. Stein. Better online buffer management. In *Proceedings 18th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 199–208, 2007.

[29] O. Mutlu and T. Moscibroda. Stall-time fair memory access scheduling for chip multiprocessors. In *Proceedings 40th Annual IEEE/ACM International Symposium on Microarchitecture*, pages 146–160, 2007.

[30] H. Räcke, C. Sohler, and M. Westermann. Online scheduling for sorting buffers. In *Proceedings 10th Annual European Symposium on Algorithms*, pages 820–832, 2002.

[31] S. Rixner, W. J. Dally, U. J. Kapasi, P. R. Mattson, and J. D. Owens. Memory access scheduling. In *Proceedings 27th Annual International Symposium on Computer Architecture*, pages 128–138, 2000.

[32] M. Schmidt. Packet buffering: Randomization beats deterministic algorithms. In *Proceedings 22nd Annual Symposium on Theoretical Aspects of Computer Science*, pages 293–304, 2005.

A   Why Some Natural Strategies Fail?

In the following, we study two natural greedy algorithms, and prove that both of them fail to attain near-optimal throughput even when $C = o(L)$. In particular, we establish that neither algorithm has a competitive ratio better than $1/2$.

We begin by considering a natural modification of the EDF strategy to our setting. We refer to this modified strategy as MEDF. At any point in time, it transmits a pending packet with minimal deadline time. In case that the packet to be transmitted has a different color than the current active color then it is discarded, and the corresponding time-slot is dedicated to a color transition. When there are several pending packets with minimal deadline time, MEDF arbitrarily transmits one of them, say a packet with a "minimal color". The following lemma establishes that MEDF is an *optimal* $1/2$-competitive online algorithm for the general problem, i.e., when no assumptions are made with respect to the input instances. Note that the optimality of MEDF follows from Theorem 4.1, which argues that no deterministic online algorithm can achieve a competitive ratio better than $1/2$.

Lemma A.1. *MEDF attains a competitive ratio of $1/2$.*

Proof. Consider some input sequence σ. Let ALG denote the throughput of the schedule generated by MEDF with respect to σ, and let ALG$'$ be the throughput of the EDF schedule with respect to σ, that is, the schedule generated when one assumes that all packets have the same color. Moreover, let OPT be the throughput of the optimal schedule. It is clear that ALG$' \leq 2$ALG since MEDF uses the EDF schedule, and drops no more than one packet in any two consecutive time-slots. Now, recall that EDF generates schedules having optimal throughput with respect to input instances in which all packets have the same color. This implies, in conjunction with the simple fact that any schedule for a given instance is a valid schedule for the same instance in which all colors are identical, that OPT $\leq$ ALG$'$. ∎

We now establish that even if one only considers input instances having $C = o(L)$, MEDF cannot achieve a competitive ratio better than $1/2$.

Theorem A.2. *The competitive ratio of MEDF cannot be better than $1/2$, even when $C = o(L)$.*

Proof. Let us consider an input instance in which the laxity of all packets is identical. Specifically, the packets sequence is $\sigma = \langle (1, c_1), (1, c_2), \ldots, (L, c_1), (L, c_2) \rangle$, where a pair (r, c) stands for a packet whose release time is r, its color is c, and its deadline is $r + L$. Figure 3 provides a schematic description of the schedule generated by MEDF, and the optimal schedule. In particular, MEDF generates an initial EDF schedule in which it alternately transmits a packet with color c_1 and a packet with color c_2. However, due to the color transition requirement, every second packet is dropped, and thus its throughput is L. On the other hand, an optimal schedule can transmit all packets except one, and yield a throughput of $2L - 1$. This implies that the competitive ratio of MEDF cannot be better than $1/2 + 1/(4L - 2)$. Notice that one may select any $L \to \infty$, and thus the theorem follows. ∎
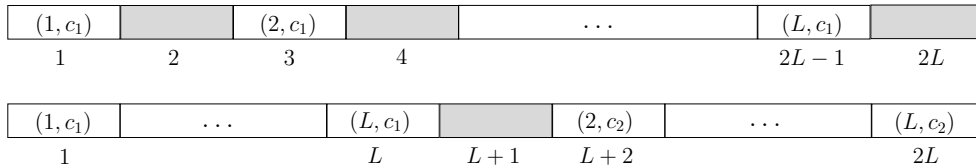

Figure 3: The schedule generated by MEDF (above), and an optimal schedule (below). Note that grayed boxes indicate non-productive time-slots (e.g., color transition time-slots).

Essentially, the main drawback of MEDF is that it may generate schedules in which many packets are dropped. We proceed by considering the *color-greedy* (CG) strategy. At any point in time, this strategy transmits a pending packet whose color is identical to the current active color. It is clear that this strategy addresses the main drawback of MEDF. However, as demonstrated below it cannot be better than $1/2$ competitive even when $C = o(L)$.

Theorem A.3. *The competitive ratio of CG cannot be better than $1/2$, even when $C = o(L)$.*

Proof. Similarly to the proof of Theorem A.2, consider an input instance having the same laxity for all packets. Specifically, the packets sequence is $\sigma = \langle (1, c_2), (2, c_1)^{L-2}, (2, c_2), \ldots, (L + 2, c_2) \rangle$. Here, the notation $(r, c)^k$ stands for k packets of the type determined by the pair (r, c). Figure 4 provides a schematic description of the schedule generated by CG, and the optimal schedule. Essentially, CG repetitively transmits packets with color c_2, and therefore fails to transmit any of the packets with color c_1 before their deadline. Accordingly, its throughput is $L + 2$. On the other hand, an optimal schedule can transmit all packets, and attain a throughput of $2L$. This implies that the competitive ratio of CG cannot be better than $1/2 + 1/L$. Note that one may select any $L \to \infty$, and hence the theorem follows. ∎

B   NP-hardness of the Offline Problem

In what follows, we show that the offline version of buffer management for colored packets with deadlines in which the sequence of packets is known in advance is NP-hard. To this end, we exhibit a reduction from 3-Partition, as defined below, to the buffer management problem.
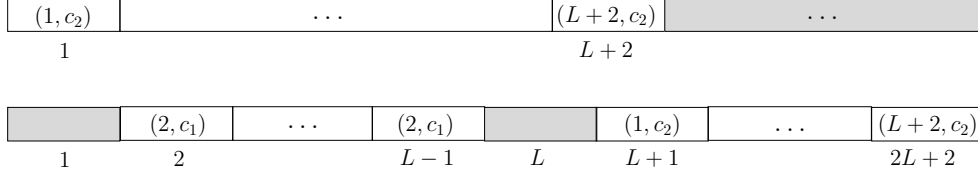
Figure 4: The schedule generated by CG (above), and an optimal schedule (below). Note that grayed boxes indicate non-productive time-slots.

Definition B.1. (3-Partition) Given a multiset $S = \{a_1, a_2, \ldots, a_n\}$ of $n = 3m$ positive integers, determine if S can be partitioned in to m subsets $S_1, S_2, \ldots, S_m$ each of size 3 such that sum of the numbers in each subset is equal.

Theorem B.2. ([19, 18]) *3-Partition is strongly NP-complete, i.e, NP-complete even when the integers $\{a_1, a_2, \ldots, a_n\}$ are bounded by a polynomial in n. Let $A = \sum_{i=1}^{n} a_i$ and $B = A/m$. The problem remains NP-complete even if each number a_i is restricted to $(B/4, B/2)$.*

Using a reduction from the 3-Partition problem, we obtain the following NP-hardness result.

Theorem B.3. *Given a sequence $\sigma = \{(r_i, d_i, c_i)\}$ of packets characterized by release time r_i, deadline time d_i, and color c_i, it is NP-hard to determine if all the packets in the sequence can be transmitted during their respective time frames $[r_i, d_i]$. Further, the problem remains NP-hard even if the laxities of all packets are equal.*

Proof. Let $S = \{a_1, a_2, \ldots, a_n\}$ be an instance of 3-Partition with $n = 3m$. Recall that $A = \sum_{i=1}^{n} a_i$ and $B = A/m$. Note that A is divisible by m, since otherwise the 3-Partition instance S is trivially infeasible. Set $D = (B + 6)m$. Construct a packet sequence σ with the following packets. Observe that the laxity of all packets is exactly D.

- **Dummy Packets:** $D - 1$ packets with parameters $(0, D, \mathsf{Red})$ and $D - 1$ packets with parameters $(2D - 1, 3D - 1, \mathsf{Blue})$. The intent of Red and Blue packets is to occupy all the time-slots between $(0, D - 1)$ and $(2D, 3D - 1)$ respectively.

- **Bin Markers:** m pairs of packets $\{P_i, P_i' : 0 \leq i \leq m - 1\}$ with

$$
\begin{aligned}
P_i &= (i \cdot (B + 6) + 1, i \cdot (B + 6) + D + 1, C_i) \\
P_i' &= (i \cdot (B + 6) + D + 1, i \cdot (B + 6) + 2D + 1, C_i)
\end{aligned}
$$

  Note that the pairs P_i, P_i' are of the same color, and the release time of P_i' matches the deadline of P_i. Thus, a pair of packets $\{P_i, P_i'\}$ can be transmitted consecutively only if P_i is transmitted at its deadline, and P_i' is transmitted in the next time-slot. If every pair $\{P_i, P_i'\}$ is transmitted in consecutive time-slots, then it would divide the remaining time-slots between $[D + 1, 2D]$ in to intervals (bins) of length $B + 4$. Hence, these packets are referred to as bin markers.

- **Main Packets:** For each number a_i in the 3-Partition instance S, introduce a_i packets with parameters $(D, 2D, c_i)$.

14

Completeness Let $S_1, S_2, \ldots, S_m$ denote a feasible 3-Partition of the instance S with the i-th set S_i is given by $S_i = \{a_{i_1}, a_{i_2}, a_{i_3}\}$. It is easy to see that the following is a feasible schedule that transmits all the packets of the sequence σ within their deadline.

- Transmit the Red packets in the first $D - 1$ time-slots, followed by a color transition time-slot.

- For $i = 0$ to $m - 1$ do

    - Transmit packets P_i, P'_i in time-slots $i \cdot (B + 6) + D, i \cdot (B + 6) + D + 1$, followed by a color transition time-slot.
    - In the $B + 3$ time-slots from $i \cdot (B + 6) + D + 3$ to $(i + 1) \cdot (B + 6) + D$, Transmit the a_{i_1} packets of color c_{i_1}, perform a color transition, transmit a_{i_2} packets of color c_{i_2}, perform a color transition, and finally transmit a_{i_3} packets of color c_{i_3}, followed by a color transition.

- Transmit the Blue packets in the $D - 1$ time-slots between $(2D, 3D - 1)$

Soundness Let us suppose there is a feasible schedule to transmit all the packets in the sequence σ. The total number of packets in the sequence σ is given by

$$\text{Number of Packets}(\sigma) = 2 \cdot (D - 1) + 2m + \sum_i a_i = 2D - 2 + 2m + Bm = 3D - 4m - 2$$

The number of colors in the sequence σ is given by $2 + m + n = 4m + 2$. Recall that in order to transmit packets of k different colors, the schedule requires at least $k - 1$ color transitions. Hence, any feasible schedule that transmits all packets require at least $4m + 1$ color transitions. This implies that any schedule that transmits all packets requires at least $3D - 4m - 2 + 4m + 1 = 3D - 1$ time-slots. Now, notice that the latest deadline in the packet sequence is $3D - 1$. Therefore, a feasible schedule that transmits all packets necessarily uses exactly $4m + 1$ color transitions. Consequently, we can conclude that the schedule transmits all packets of the same color together.

This implies that the schedule transmits the bin marker pairs $\{P_i, P'_i\}$ of color C_i together. By our choice of release and deadline times of P_i, P'_i, this forces P_i to be transmitted in time-slot $i \cdot (B + 6) + D$ and P'_i in time-slot $i \cdot (B + 6) + D + 1$. The $D - 1$ Red packets are to be scheduled together before the deadline of D. Due to the scheduling of packets P_0, P'_0, time-slot D is used for a color transition. Hence, the Red packets are all scheduled in the first $D - 1$ time-slots. For each $0 \le i \le m - 2$, there are $B + 4$ time-slots between the transmissions of bin markers $\{P_i, P'_i\}$ and $\{P_{i+1}, P'_{i+1}\}$. The time-slot after the transmission of $\{P_i, P'_i\}$ and before the transmission of $\{P_{i+1}, P'_{i+1}\}$ are used for color transitions. Consider the remaining $B + 2$ free time-slots between bin markers $\{P_i, P'_i\}$ and $\{P_{i+1}, P'_{i+1}\}$. The only packets that can be transmitted in these time-slots are the main packets of color c_i for some i. Recall that all the a_i packets of color c_i are transmitted together, and each a_i is strictly between $B/4$ and $B/2$. Hence, packets of at most three different colors are transmitted in these $B + 2$ time-slots. Furthermore, each of these $B + 2$ time-slots is used either for a transmission or a color transition. Thus, exactly three different colors $c_{i_1}, c_{i_2}, c_{i_3}$ are transmitted, with two color transitions needed in between. This would also imply that $a_{i_1} + a_{i_2} + a_{i_3} = B$. Finally, for $0 \le i \le m - 2$, if $c_{i_1}, c_{i_2}, c_{i_3}$ are the colors of main packets transmitted between bin markers $\{P_i, P'_i\}$ and $\{P_{i+1}, P'_{i+1}\}$, then let $S_{i+1} = \{a_{i_1}, a_{i_2}, a_{i_3}\}$. Let $S_m = S - \cup_{i=1}^{m-1} S_i$. From the above argument, it is clear that the sets $\{S_i\}$ form a feasible 3-Partition of the instance S. ∎