An optimal online algorithm for packet scheduling with agreeable deadlines

Fei Li* Jay Sethuraman† Clifford Stein‡

1 Introduction

An important issue in IP-based QoS networks is the effective management of packets at the router level. Specifically, if the arriving packets cannot all be stored in a buffer, or if the packets have deadlines by which they must be delivered, the router needs to identify the packets that should be dropped. In recent work, Kesselman et al. [6] propose a model, called *buffer management with bounded delay*, which can be thought of as an online scheduling problem on a single machine: packets arrive at a network switch and are stored in a buffer of size B. Each packet has a positive weight and a deadline, with the weight representing the value of transmitting the packet by its deadline. At each integer time step, exactly one packet can be transmitted, and the objective is to maximize the total weight of the transmitted packets. If $B = \infty$, this is the online version of the scheduling problem $1|\ p_j = 1, r_j, d_j\ |\sum w_j U_j$. (We assume that r_j and d_j are integers.)

Previous work. A simple greedy algorithm that always schedules a maximum-weight available job is 2-competitive [5, 6]. An improved algorithm [3] is ≈ 1.939-competitive. A lower-bound of $\phi := (\sqrt{5}+1)/2$ for deterministic algorithms is known [5, 4, 1], as is a randomized algorithm with a competitive ratio of $e/(e-1) \approx 1.582$, and a lower bound of 1.25 [2].

Several restricted variants have been considered. Define the *span* of a job to be the difference between its deadline and release date. An instance is *s-bounded* if the span of any job is at most s, and *s-uniform* if the span of any job is exactly s. These cases arise naturally in the buffer management problem in which only the end-to-end delay matters, and only a small amount of delay can be tolerated. If for any two jobs i and j, $r_i < r_j$ implies $d_i \leq d_j$, then we say they have *agreeable* deadlines. Note that s-uniform instances are a special case of the instances with agreeable deadlines. The lower-bound construction in [5, 4, 1] is valid for both 2-bounded instances as well as for instances with agreeable deadlines. For s-bounded instances, Bartal et al. [2] propose an algorithm with competitive ratio $2 - 2/s + o(1/s)$; this algorithm has (the best possible) competitive ratio ϕ for $s = 2, 3$, and $\sqrt{3}$ for $s = 4$. For instances with agreeable deadlines (and hence also s-uniform instances), Chrobak et al. [3] propose an algorithm with competitive ratio ≈ 1.838.

Our result. Our main result is an online algorithm with competitive ratio ϕ for instances with agreeable deadlines. Both the algorithm and the analysis are strikingly simple. Note that this competitive ratio is the best achievable by a deterministic algorithm.

2 Algorithm and Analysis

We consider the problem of *agreeable* deadlines. A buffer of size B stores all the packets that have not been dropped yet. We allow *early dropping* of packets; thus if three packets arrive, each with span 2, the algorithm can immediately drop a minimum-weight packet. At each time step, the online algorithm needs to transmit a packet from the buffer, without any knowledge of future arrivals. Recall that our objective is to find an online algorithm that maximizes the *total value* or *gain* (sum of the values) of the transmitted packets. We assume the packets in the buffer are in a *canonical* order: increasing deadline, with ties broken in order of decreasing weight.

Algorithm. Why does the greedy algorithm fail to be better than 2-competitive? Let (w, d) denote a packet with weight w and deadline d. Then, faced with 2 packets $(1-\epsilon, 1)$ and $(1, 2)$, the greedy algorithm chooses the latter, even though the former has only slightly lower value. A variant is to choose the earliest packet among a set of sufficiently large weight, but this runs into a similar difficulty. Our idea (called MG for modified greedy) is to identify a packet that has a "sufficiently large" weight compared to the maximum-weight packet, but also a "sufficiently large" weight compared to an earliest-deadline packet. Formally, let e denote the first (i.e. earliest-deadline) packet in the buffer and let h denote the first maximum-weight packet in the buffer. Note that e has the maximum weight among all earliest deadline packets; and h has the earliest deadline among all maximum-weight packets. Also, a set of packets is *feasible* if each packet can be scheduled before its deadline.

Online algorithm MG works as follows: at the

*Dept. of CS, Columbia U. lifei@cs.columbia.edu

†Dept. of IEOR, Columbia U. jay@ieor.columbia.edu Research partially supported by NSF Grant DMI-0093981

‡Dept. of IEOR, Columbia U. cliff@ieor.columbia.edu Research partially supported by NSF Grant DMI-9970063

beginning of each time step t, MG considers the packets in the buffer and the newly arrived packets, and from this set selects the maximum-valued feasible subset of at most B packets. The remaining packets are dropped, and the packets in the buffer are arranged in canonical order. (This can be computed efficiently via matching.) The algorithm then identifies the packets e and h. If $w_e \geq w_h/\phi$, then e is sent; otherwise the earliest packet f satisfying both $w_f \geq \phi w_e$ and $w_f \geq w_h/\phi$ is sent. Such a packet exists because h itself is a candidate; so either $f = h$ or f is some packet in the buffer that appears between e and h. In particular, $d_f \leq d_h$.

Analysis. Our analysis is essentially a potential function argument. Suppose that at some time t, MG and the adversary have identical buffers. Both MG and the adversary will each process arriving packets and transmit a packet, but at the end of the time step, their buffer contents may be different. We will then modify the adversary's buffer and make it identical to the algorithm's buffer; but to do so, we may have to let the adversary collect additional weight. The crux of the analysis is to show that the algorithm's gain in this time step is at least $1/\phi$ of the adversary's *modified gain*, i.e. the sum of the weight of the packet that the adversary transmitted and the additional weight given to the adversary in modifying its buffer.

THEOREM 2.1. *Algorithm MG is ϕ-competitive.*

Proof Sketch We may assume wlog that the adversary (but not MG) delivers packets in non-decreasing deadline order. Fix a time t, and suppose MG and ADV have identical buffers. After processing the arrivals at time t, the buffer contents of MG and ADV will remain the same. Moreover, the packets that arrived at time t and stored in the buffer (if any) will form the tail of the buffer. The proof proceeds by considering various cases, depending on which packet MG transmits. Let v_{ADV} and v_{MG} denote the value collected by the (modified) adversary and the algorithm at time t.

1) If ADV and MG transmit the same packet, $v_{ADV} = v_{MG}$ and their buffers are identical.

2) Suppose MG transmits e, and ADV transmits a packet j. Since $w_e \geq w_h/\phi \geq w_j/\phi$, $v_{ADV} = w_j \leq \phi w_e = \phi v_{MG}$. Modify the adversary's buffer by replacing packet e with packet j; this only helps the adversary. After this step, ADV and MG have identical buffers, and $v_{ADV}/v_{MG} \leq \phi$, as required.

3) Suppose MG transmits $f \neq e$ and ADV sends e. Note that $w_f \geq \phi w_e$, and $d_f > d_e$. Clearly, ADV must transmit f eventually, if not it gains by transmitting f now instead of e. We let ADV send both e and f in this time step and keep e in its buffer. Now, $v_{ADV} = w_e + w_f$, and $v_{MG} = w_f$. Since $w_f \geq \phi w_e$, $v_{ADV}/v_{MG} = 1 + w_e/w_f \leq 1 + 1/\phi = \phi$.

4) Suppose MG sends packet $f \neq e, h$, and ADV sends a packet j that is after packet f in the buffer. Since ADV transmits packets from the buffer in deadline order, $d_j > d_f$, ADV does not send f, and so $w_j > w_f$.

As $w_f \geq w_h/\phi \geq w_j/\phi$, $v_{ADV} = w_j \leq \phi w_f = \phi v_{MG}$. The adversary's buffer does not contain j, but contains f; and MG's buffer contains j, but not f. To make ADV's buffer identical to MG's, we increase f's value to w_j and increase its deadline to d_j. This helps the adversary and $v_{ADV}/v_{MG} \leq \phi$.

5) Suppose MG sends packet $f \neq e$ and ADV sends a packet $j \neq e$ that is earlier than f in the buffer. Clearly, $w_j < w_f$ and $d_j < d_f$; moreover $w_j, w_f > w_e$, $d_j, d_f > d_e$, and e is dropped by the adversary (because of EDD order). We know that f must be eventually transmitted by the adversary. We argue next that the adversary has a feasible schedule in which f is transmitted now, regardless of the future arrivals.

For convenience, let $p_1, p_2, \ldots,$ be the packets in the buffer; note that $f = p_l$, $j = p_k$, for some $k < l$, $k, l \neq 1$; and $e = p_1$. Since the packets are all schedulable in the absence of future arrivals, $d_{p_i} \geq t + i$. A packet p_i is *critical* if $d_{p_i} = t + i$. The adversary schedules p_k and p_l, and possibly some packets that appear in between. We also know that the adversary drops $e = p_1$. Since $d_{p_i} \geq t + i$, and since the future arrivals all have deadlines no earlier than the deadline of f, none of the packets $p_k, p_{k'}, \ldots, p_{l'}, p_l$ transmitted by the adversary is critical. So the sequence $p_l, p_k, p_{k'}, \ldots, p_{l'}$ is a valid transmission sequence for the adversary.

So we may assume that f is sent by the adversary during this time slot. So MG and ADV transmit the same packet and gain identical weight in this time step.

References

[1] N. Andelman, Y. Mansour, and A. Zhu, *Competitive Queuing Polices for QoS Switches*, SODA 2003.

[2] Y. Bartal, F. Y. L. Chin, M. Chrobak, S. P. Y. Fung, W. Jawor, R. Lavi, J. Sgall, and T. Tichy, *Online Competitive Algorithms for Maximizing Weighted Throughput of Unit Jobs*, 21st Annual Symposium on Theoretical Aspects of Computer Science, pp.190 – 210, 2004.

[3] M. Chrobak, W. Jawor, J. Sgall, and T. Tichy, *Improved Online Algorithms for Buffer Management in QoS Switches*, ESA 2004.

[4] F. Y. L. Chin and S. P. Y. Fung, *Online Scheduling with Partial Job Values: Does Timesharing or Randomization Help?*, Algorithmica, 2003.

[5] B. Hajek, *On the Competitiveness of Online Scheduling of Unit-Length Packets with Hard Deadlines in Slotted Time*, Proceedings of 2001 Conference on Information Sciences and Systems, The John Hopkins U., 2001.

[6] A. Kesselman, Z. Lotker, Y. Mansour, B. P. Shamir, B. Schieber, and M. Sviridenko, *Buffer Overflow Management in QoS Switches*, STOC 2001, pp.520 – 529.