Forwarding Packets Greedily*

Joan Boyar
University of Southern Denmark
joan@imada.sdu.dk

Lene M. Favrholdt
University of Southern Denmark
lenem@imada.sdu.dk

Kim S. Larsen
University of Southern Denmark
kslarsen@imada.sdu.dk

Kevin Schewior
University of Cologne
k.schewior@uni-koeln.de

Rob van Stee
University of Siegen
rob.vanstee@uni-siegen.de

March 5, 2026

Abstract

We consider the problem of forwarding packets arriving online with their destinations in a line network. In each time step, each router can forward one packet along the edge to its right. Each packet that is forwarded arrives at the next router one time step later. Packets are forwarded until they reach their destination. The flow time of a packet is the difference between its release time and the time of its arrival at its destination. The goal is to minimize the maximum flow time.

This problem was introduced by Antoniadis et al. in 2014. They propose a collection of natural algorithms and prove for one, and claim for others, that none of them are $O(1)$-competitive. It was posed as an open problem whether such an algorithm exists.

We make the first progress on answering this question. We consider the special case where each packet needs to be forwarded by exactly one or two routers. We show that a greedy algorithm, which was not previously considered for this problem, achieves a competitive ratio of exactly $2 - 2^{1-k}$, where k is the number of active routers in the network. We also give a general lower bound of 4/3, even for randomized algorithms.

1 Introduction

We consider packet scheduling in a network. Routers in networks often have a backlog of packets waiting to be sent, and it is important to have a good policy for choosing which packet to forward at any given time. Generally we would like to optimize the Quality of Service (QoS) to the application-layer clients. However, this is difficult or impossible to achieve as routers are generally not aware of which packet is being sent by which client. As in Antoniadis et al. [3], we therefore focus on the next best option which is to optimize the QoS of the packets themselves, in the form of minimizing their flow time.

*The first four authors were supported in part by the Independent Research Fund Denmark, Natural Sciences, grants DFF-0135-00018B and DFF-4283-00079B.

More specifically, the problem we consider is online packet forwarding, where the routers are in a simple network, the line. Packets are released in an online manner, each with an origin and destination router. The model is synchronous. In each time step, any number of packets can be released on any number of routers, and any router can forward at most one packet to its neighbor to the right. A router's buffer for incoming packets has unbounded capacity. The routers do not know the status of any other routers. The object is to minimize the maximum flow time, where the flow time of a packet is the time between its release time and its arrival at its destination. We restrict the problem to packets that need to be forwarded by at most two routers, since, as detailed below, the core challenges show up already when dealing with such packets.

We measure algorithms by their competitive ratios. Let OPT be an optimal offline algorithm for packet routing on the line. For any algorithm, ALG, we let $\text{ALG}(I)$ denote the maximum flow time in ALG's solution for the input sequence I. For any $c \geq 1$, an online algorithm, ALG, is said to be c-competitive, if there exists a constant b such that for any input sequence I, $\text{ALG}(I) \leq c \cdot \text{OPT}(I) + b$. The (asymptotic) competitive ratio of ALG is the infimum value c for which ALG is c-competitive.

This work is motivated by the results by Antoniadis et al. [3], who defined this appealingly clean but "surprisingly challenging" (a predicate we tend to agree with) problem. Antoniadis et al. showed that a natural algorithm, Earliest Arrival, is not $O(1)$-competitive, and they also claimed for *three* other natural *classes of* algorithms that they could not be $O(1)$-competitive, seemingly leaving no natural algorithm as a candidate for being $O(1)$-competitive. They posed the existence of an $O(1)$-competitive algorithm as an open problem, on the outcome of which they revealed there was a modest wager. In this work, the first work on this problem since the publication more than a decade ago, we consider the natural GREEDY algorithm. While we do not settle the wager, we do contribute to the understanding of this problem and leave GREEDY as a natural candidate for being $O(1)$-competitive.

Contributions There seems to be a fundamental trade-off in maximum flow time on the line between the following two proxy objectives, both vaguely aligned with minimizing the maximum flow time:

- Give highest priority to packets with the earliest release times. This is what the algorithm Earliest Arrival does, which is not $O(1)$-competitive [3], essentially because it disregards how far the packets still have to travel.
- Focus on precisely the latter: Prioritize packets by how far they still have to travel (their remaining *length*). This is what the algorithm Furthest-To-Go does, about which Antoniadis et al. claim that it is not $O(1)$-competitive [3]. Indeed, consider a length-1 packet that is repeatedly delayed by a continuous stream of length-2 packets but could have just been forwarded first, at maximum flow time 3. In other words, the algorithm fails because it disregards when packets were released, namely the first proxy objective.

In this work, we formalize that there *is* a non-trivial trade-off to be solved by providing the first *general* lower bound on the achievable competitive ratio. In Section 4, we show that even randomized algorithms are not better than 4/3-competitive. In our construction, we iteratively force the algorithm to make an (unfortunately wrong) decision on how to solve the above trade-off. Remarkably, the lower bound only requires packets of length 1 or 2.

We propose an algorithm that we term GREEDY, which solves the aforementioned trade-off in an elegant way: Simply prioritize packets according to the sum of the two proxy objectives, namely

how long they have been around plus the distance they still have to go. (We give a different, equivalent formulation later.) In yet other words, this algorithm prioritizes the packets by the flow time they are going to get under the optimistic assumption that they are not delayed any further—therefore the name. So GREEDY does not act by a "proxy objective" but by the *real* objective. Viewed this way, the reader may agree that this is the most natural policy to try.

Our main result is an analysis of Greedy in the case mentioned above—packets of length 1 or 2. While this may seem restrictive, recall that the fundamental trade-off already shows up in this case. With a relatively short but sophisticated proof, we show that GREEDY is $(2 - \frac{1}{2^{k-1}})$-competitive, where k is the number of active routers in the line network. We also prove that the result is tight by proving a matching lower bound for GREEDY. These results are presented in Section 3, where we start with the lower bound to build some intuition.

We conjecture that the competitive ratio of GREEDY is constant, possibly even 2. Indeed, we do not see how longer packets can be utilized to increase the lower bound on GREEDY further. If our conjecture is true, we believe that our techniques will prove useful in establishing this result.

Related work Most previous work on routing algorithms under the adversarial model has focused on questions of stability (will the number of packets in the system remain bounded over time?) and throughput maximization. We refer to [2, 7, 9, 17, 19, 20] for a selection of papers on stability and [4, 6, 10, 14, 16, 18, 22] for papers on throughput maximization.

In the aforementioned work [3], Antoniadis et al. show that Earliest Arrival and Furthest-To-Go are scalable for the maximum-flow-time objective, i.e., $O(1)$-competitive with arbitrarily small speed augmentation. They also show that there is no $O(1)$-competitive algorithm for average flow time. Havill [13] considered online packet routing on bidirectional paths and rings with the objective of minimizing the makespan. For online algorithms, they showed that Longest In System and Moving Priority have competitive ratio 2 on linear arrays and also gave a 2-competitive algorithm for bidirectional ring routing. Im and Moseley [15] considered speed scaling for job scheduling on machines that are connected via a tree to minimize the total flow time and gave constant-competitive algorithms with constant speed augmentation. Disser et al. [11] give approximation algorithms for bidirectional scheduling on a path to minimize the total completion time and show that this problem is NP-hard. Erlebach and Jansen [12] study the problem of establishing and completing a given set of calls as early as possible for bidirectional and directed calls in various classes of networks. Adamy et al. [1] studied call control in rings. The goal is to compute, for a given ring network with edge capacities and a set of paths in the ring, a maximum cardinality subset of the paths such that no edge capacity is violated. They give a polynomial-time algorithm to solve the problem optimally.

The problem we consider can be viewed as a special case of job shop scheduling with unit processing times [5] where each job needs to be processed on a contiguous set of machines in left-to-right order. Soukhal et al. [21] considered flow shop on two machines. Wei and Yuan [23] considered two-machine flow shop scheduling with equal processing times. See also Chen [8].

2 Preliminaries

We let k denote the number of active routers in the line network, i.e., those that have a neighbor to the right that they can forward packets to. Thus, the last router on the line, which can only receive packets, is ignored. Similarly, if a router i is the last router that needs to forward a packet

p, we say that router i is the *last router* of p, even though the *destination* of p is router $i+1$. When a router forwards a packet, we also say that it *processes* the packet.

The release time of a packet p is denoted $r(p)$. The *length*, $\ell(p)$, of p is the number of routers that need to process p, so $\ell(p) \in \{1, 2\}$. At any given time $t \geq r(p)$, p's remaining length (the number of routers that still need to process p) is denoted $\ell(p, t)$. The *completion time*, $C(p)$, of a packet p is the first time t such that $\ell(p, t) = 0$, and its *flow time* is $C(p) - r(p)$.

We define the *delay* of packet p at time t by

$$d(p, t) = t - r(p) - (\ell(p) - \ell(p, t))$$

and the *priority* of packet p at time t by

$$\pi(p, t) = \ell(p) + d(p, t) = t - r(p) + \ell(p, t).$$

The delay of a packet is hence the number of time steps it has been in the system minus the number of time steps that it has been processed so far. Note that this means that once the priority of a packet is the flow time it will finally have, the packet is processed in every time step from that point on until it is done.

Without loss of generality, we assume that Opt is *zealous* on each router, meaning that if there is at least one packet to be processed on a router, then Opt processes a packet.

3 The Algorithm Greedy

Since the priority of a packet at any point in time is a lower bound on its flow time, a natural greedy choice would be to forward packets with high priority. This is what the greedy algorithm does.

Greedy: In each time step, on each router where at least one packet is waiting, forward a packet with the highest priority on that router. We remark that an arbitrary tie-breaking rule may be used.

By providing matching upper and lower bounds, we show that the competitive ratio of Greedy is exactly $2 - 1/2^{k-1}$ (recall that k is the number of active routers in the network).

3.1 Lower bound for Greedy

We start with the lower bound on Greedy's competitive ratio, gathering some intuition from concrete examples before proving the matching upper bound. Note that for only one router, Greedy is optimal, since both it and Opt are zealous. As a warm-up to the general result (Theorem 2), we first consider $k = 2$ routers, using the same notation as in the proof of Theorem 2.

Proposition 1 The competitive ratio of Greedy is at least $3/2$, even with only $k = 2$ routers.

Proof Let $h \geq 4$. The following packets are released in sequence I:

- There are h packets released at time 0 that need to be processed on router 1. This set of packets is denoted A_1.
- There are h packets released at time 2 that need to be processed on routers 1 and 2. This set of packets is denoted B_1.

- There are $2h$ packets released at time $h+3$ that need to be processed on router 2. This set of packets is denoted B_2.

Since the packets in B_1 are released two time steps later than those in A_1, GREEDY prioritizes packets in A_1 on router 1. On router 2, GREEDY prioritizes the packets in B_1 over those in B_2. Thus, GREEDY processes B_1 on router 1 in time steps h to $2h-1$ and on router 2 in time steps $h+1$ to $2h$. Therefore, the packets in B_2 are processed in time steps $2h+1$ to $4h$, resulting in a maximum flow time of $3h-2$.

An optimum schedule, in contrast, could achieve a maximum flow time of $2h$ by giving priority to the packets in B_1 on router 1, thus processing the packets in A_1 in time steps 0 to 1 and $h+2$ to $2h-1$, processing the packets in B_1 on router 1 in time steps 2 to $h+1$ and on router 2 in time steps 3 to $h+2$, and processing the packets in B_2 in time steps $h+3$ to $3h+2$. Thus, the flow time of the last packet in A_1 is $2h$ and for B_2 it is also $2h$.

Hence, $\text{GREEDY}(I) = \frac{3}{2}\,\text{OPT}(I) - 2$, giving a lower bound of $\frac{3}{2}$ on GREEDY's competitive ratio. □

More generally, we extend this construction, considering k routers, where $k \geq 2$.

Theorem 2 For k routers, the competitive ratio of GREEDY is at least $2 - \frac{1}{2^{k-1}}$.

Proof First, we define the request sequences, with parameters k and $h \geq 2$, where h can be arbitrarily large and is necessary to make the result asymptotic. The sequence I_k^h contains the blocks of packets $A_1^h, A_2^h, \ldots, A_{k-1}^h$ and $B_1^h, B_2^h, \ldots, B_k^h$ defined below. In the following, we omit the superscript h on the A_i^h and B_i^h blocks, since it is always h.

All packets have length 2, except those in A_1 and B_k, which have length 1. This is done to strengthen the lower bound (improve the dependence on the number of routers k). The release times of the packets in blocks A_i and B_i depend on the total number of packets in $B_1, B_2, \ldots, B_{i-1}$, based on

$$r_i = \left(\sum_{j=1}^{i-1} |B_j| \right) + \max\{0, i-2\}.$$

For each block, all packets in it will be released at the same time. Let $r(X)$ denote the release time for the packets in a block X. For $1 \leq i \leq k-1$:

- Block A_i: $2^{k-1-i} \cdot h$ packets released on router $\max\{1, i-1\}$ at time $r(A_i) = r_i$.
- Block B_i: $(2^{k-1} - 2^{k-1-i})h$ packets released on router i at time $r(B_i) = r_i + 2$.

Block B_k: $2^{k-1} \cdot h$ packets released on router k at time $r(B_k) = r_k + 2$.

In order to consider and compare the largest flow times for GREEDY and OPT, respectively, we use the following equations that hold for $1 \leq i \leq k-1$:

$$|A_i| + |B_i| = |B_k| = 2^{k-1} \cdot h \tag{1}$$

$$\sum_{j=1}^{i} |A_j| = \sum_{j=1}^{i} 2^{k-1-j} \cdot h = \left(2^{k-2} + \cdots + 2^{k-1-i}\right) h = \left(2^{k-1} - 2^{k-1-i}\right) h = |B_i| \tag{2}$$

For the case of $k = 4$ routers, GREEDY's and OPT's schedules are depicted in Figure 1. First, we argue that GREEDY's schedule is as shown.

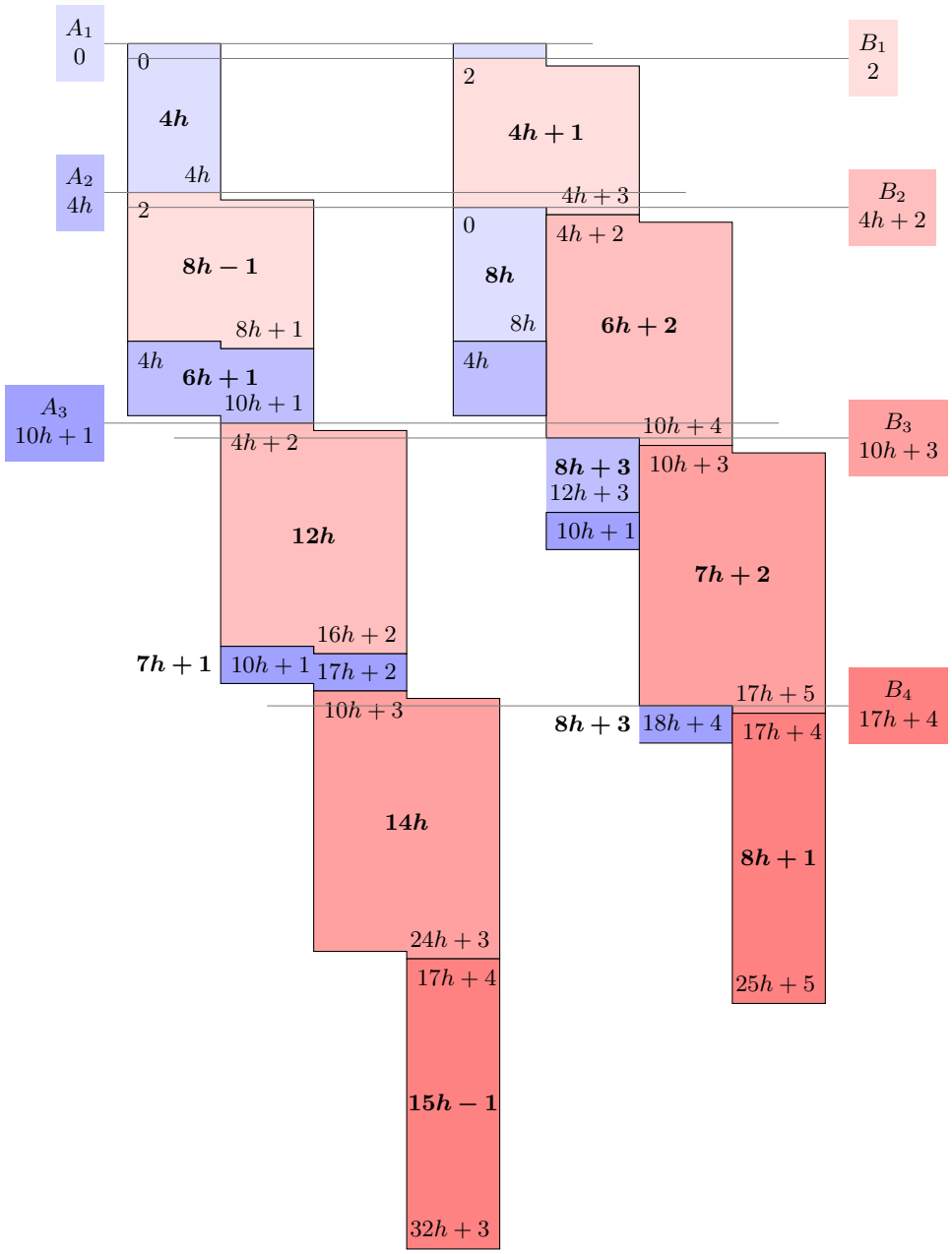

Figure 1: Example of the lower-bound construction for the number of routers $k = 4$. Outermost, to the left and right, we indicate the release times for the different blocks (the placement to the left or right is only for readability and bears no algorithmic significance). The lines from the little boxes with the names and release times show where these release times are. Innermost, GREEDY's schedule is on the left and OPT's on the right. A block that is open to the left indicates a continuation of the processing of a block on the next router. Release times of the blocks are indicated at the top left, completion times at the bottom right, and flow times in the middle.

According to GREEDY's priorities, π, releasing the B-blocks 2 time steps later than the corresponding A-blocks, and releasing blocks A_i and B_i at least $|B_{i-1}|$ time steps later than A_{i-1} and B_{i-1}, respectively, we have that, at any point in time t, on any router,

- if a packet a from A_i and packet b from B_i are both waiting to be processed, $\pi(a,t) > \pi(b,t)$.
- if a packet b from B_i and packet a' from A_{i+1} are both waiting to be processed, $\pi(b,t) > \pi(a',t)$.

We later show that the first B_i packet arrives on router $i+1$ before B_{i+1} is released, so that Greedy processes packets in order A_1, B_1, A_2, B_3, dots, B_{k-1}, B_k, as depicted in Figure 1. The following calculations give the completion times and flow times for the last packet in each block for GREEDY, given that this is the order of processing.

Before the first packet in A_i is processed, all A_j and B_j blocks with $j < i$ have finished by the above. In addition, since all packets in all blocks except blocks A_1 and B_k have length 2, there is an extra time step for the packets for going from one router to the next. GREEDY finishes the last packet of A_1 at time $e_{\text{GREEDY}}(A_1) = 2^{k-2}h$, and its flow time is

$$f_{\text{GREEDY}}(A_1) = 2^{k-2}h. \tag{3}$$

For $2 \leq i \leq k-1$, GREEDY finishes the last packet of A_i by time

$$e_{\text{GREEDY}}(A_i) = \sum_{j=1}^{i} |A_j| + \sum_{j=1}^{i-1} |B_j| + i - 1. \tag{4}$$

Subtracting its release time from Equation (4), and using Equation (2), gives that its flow time is

$$f_{\text{GREEDY}}(A_i) = \sum_{j=1}^{i} |A_i| + 1 = \left(2^{k-1} - 2^{k-1-i}\right) h + 1. \tag{5}$$

For $1 \leq i \leq k-1$, GREEDY finishes the last packet of B_i by time

$$e_{\text{GREEDY}}(B_i) = \sum_{j=1}^{i} (|A_j| + |B_j|) + i, \tag{6}$$

since its start router is the router where A_i ends. Subtracting its release time from Equation (6) gives that its flow time is

$$\begin{aligned} &f_{\text{GREEDY}}(B_1) = 2^{k-1} \cdot h - 1 \text{ and} \\ &f_{\text{GREEDY}}(B_i) = \left(\sum_{j=1}^{i} |A_i|\right) + |B_i| = |B_i| + |B_i| = \left(2^k - 2^{k-i}\right) h, \end{aligned} \tag{7}$$

using Equation (2).

GREEDY finishes the last packet of B_k by time

$$e_{\text{GREEDY}}(B_k) = \sum_{j=1}^{k-1} (|A_j| + |B_j|) + |B_k| + k - 1, \tag{8}$$

since it only gets processed on one router. Subtracting its release time from Equation (8) gives that its flow time is

$$f_{\text{GREEDY}}(B_k) = \left(\sum_{j=1}^{k-1} |A_i|\right) + |B_k| - 1 = |B_{k-1}| + |B_k| - 1 = \left(2^k - 1\right) h - 1, \tag{9}$$

by Equation (2).

Now that the completion times of the packets have been calculated, we return to checking that the release time of B_{i+1} is after GREEDY has finished A_i (and therefore also B_{i-1}) is finished, ensuring that GREEDY is already busy processing B_i on router $i+1$ when B_{i+1} is released. For $1 \leq i \leq k-1$,

$$\begin{aligned} r(B_{i+1}) &= \left(\sum_{j=1}^{i} |B_j|\right) + i - 1 + 2 = |B_i| + \left(\sum_{j=1}^{i-1} |B_j|\right) + i + 1 \\ &= \left(\sum_{j=1}^{i} |A_i| + \sum_{j=1}^{i-1} |B_j|\right) + i + 1, \text{ by Equation (2)} \\ &= e_{\text{GREEDY}}(A_i) + 2, \text{ by Equation (4).} \end{aligned}$$

Finally, we note that since $r(A_{i+1}) = r(B_{i+1}) - 2$, the above implies that $e_{\text{GREEDY}}(A_i) = r(A_{i+1})$. Thus, for all $i \geq 1$, GREEDY finishes processing A_i at the release time of A_{i+1}.

We have established all the assumptions used, so Equations (4)–(9) are correct, as is the depiction of GREEDY's schedule in Figure 1, with the blocks scheduled one after another in order of their priorities. Comparing the flow times for A_i, B_i, and B_k from Equations (3), (5), (7), and (9), one sees that

$$\text{GREEDY}(I_k^h) = f_{\text{GREEDY}}(B_k) = \left(2^k - 1\right) h - 1. \tag{10}$$

Now, we turn to OPT's schedule, and, again, one can be guided by Figure 1. For the B-packets, the first packet of a block is started one time step later than its release time, and the A-blocks have a much larger delay relative to their release times. Thus, the schedule for OPT is feasible. The purpose for the adversary of releasing the B-packets earlier than what is necessary for OPT is to increase the flow time for GREEDY, since GREEDY's largest flow time is on B_k, but OPT's is on A_{k-1}.

For $1 \leq i \leq k-1$, OPT finishes the last packet of B_i by time

$$e_{\text{OPT}}(B_i) = \sum_{j=1}^{i} |B_j| + i + 2. \tag{11}$$

Subtracting its release time from Equation (11) gives that its flow time is

$$\begin{aligned} &f_{\text{OPT}}(B_1) = 2^{k-2}h + 1 \text{ and} \\ &f_{\text{OPT}}(B_i) = |B_i| + 2 = \left(2^{k-1} - 2^{k-1-i}\right) h + 2. \end{aligned} \tag{12}$$

Block B_k starts when B_{k-1} finishes, at time $\sum_{j=1}^{h-1} |B_j| + (k-1) + 2 = r(B_k) + 1$, by Equation (11). It ends after $|B_k|$ time steps, so its flow time is

$$f_{\text{OPT}}(B_k) = |B_k| + 1 = 2^{k-1}h + 1. \tag{13}$$

We now turn to the A_i blocks. OPT finishes the last packet of A_1 at time $|A_1| + |B_1| = 2^{k-1}h$. Thus, since $r(A_1) = 0$, $e_{\text{OPT}}(A_1) = f_{\text{OPT}}(A_1) = 2^{k-1}h$. For $2 \leq i \leq k-1$, OPT finishes the last packet of A_i by time

$$\begin{aligned} e_{\text{OPT}}(A_i) &= |A_i| + e_{\text{OPT}}(B_i) - 1, \text{ since } A_i \text{ ends on the router where } B_i \text{ starts} \\ &= |A_i| + \left(\sum_{j=1}^{i} |B_j| + i + 2 \right) - 1, \text{ by Equation (11).} \end{aligned} \tag{14}$$

Subtracting its release time from Equation (14) gives that its flow time is

$$\begin{aligned} f_{\text{OPT}}(A_i) &= |A_i| + \sum_{j=1}^{i} |B_j| + i + 1 - r(A_i) \\ &= |A_i| + |B_i| + 3 \\ &= 2^{k-1} \cdot h + 3, \text{ by Equation (1).} \end{aligned} \tag{15}$$

This flow time holds for any $i \geq 2$ and in particular for A_{k-1}.

Comparing Equations (12), (13), and (15), one sees that OPT's flow time for A_{k-1} is slightly larger than its flow time for B_k. Thus,

$$\text{OPT}(I_k^h) = f_{\text{OPT}}(A_{k-1}) = 2^{k-1} \cdot h + 3. \tag{16}$$

Solving for h in Equation (16), we get

$$h = \frac{\text{OPT}(I_k^h) - 3}{2^{k-1}}.$$

Then,

$$\begin{aligned} \text{GREEDY}(I_k^h) &= \left(2^k - 1\right) h - 1, \text{ by Equation (10)} \\ &= \left(2^k - 1\right) \frac{\text{OPT}(I_k^h) - 3}{2^{k-1}} - 1 \\ &= \left(2 - \frac{1}{2^{k-1}}\right) \text{OPT}(I_k^h) - 3 \left(2 - \frac{1}{2^{k-1}}\right) - 1. \end{aligned}$$

Since $\text{OPT}(I_k^h)$ grows unboundedly with h, the competitive ratio of GREEDY cannot be smaller than $2 - \frac{1}{2^{k-1}}$. □

3.2 Upper bound for Greedy

Now, we turn to the main result of the paper, the upper bound for GREEDY, beginning with some definitions. A packet, p, is *alive* for a given algorithm at any time t such that $t \geq r(p)$ and $\ell(p,t) > 0$.

Fix an optimal offline algorithm, OPT.

Definition 3 Let $a_i(t)$ be the number of alive packets in OPT's schedule that still need to be processed on router i at time t. Similarly, let $g_i(t)$ be the number of alive packets that GREEDY still needs to process on router i at time t. □

The optimal flow time is lower bounded by the maximum value of $a_i(t)$ over all time steps and over all routers.

Definition 4 We define $\Delta_i(t) = g_i(t) - a_i(t)$ for all routers i and times t. □

This means that $\Delta_i(t)$ is the number of packets that OPT is ahead on server i at time t, since it is the number of packets that OPT has already processed on router i before time t minus the number of packets that GREEDY has already processed on router i before time t. The value $\Delta_i(t)$ may be positive or negative.

Lemma 5 For each time $t \geq 0$, we have $\Delta_1(t) = 0$.

Proof Both OPT and GREEDY are zealous, and the packet arrival times are the same on router 1 for both algorithms. This means that the number of alive packets on router 1 is always the same, and $\Delta_1(t) = 0$. □

We wish to bound $\Delta_i(t)$, the number of packets that OPT is ahead on router i at time t. This is the purpose of Lemmas 6–7. The basic idea of the proof is the following. Going from router $i-1$ to router i, each unit that Δ_i increases above Δ_{i-1} corresponds to an additional packet having router $i-1$ as its last router that GREEDY has processed on $i-1$ but that OPT has not yet processed on router $i-1$. The reason that OPT did not process the packet was that it was processing another packet released on router $i-1$ that counts towards the extra packets it has processed on router i. If there have been ℓ such increases, then OPT still has to process ℓ such packets on router $i-1$. Each of these packets must have priority at least that of the packets that OPT did serve because GREEDY preferred processing them. Processing them on router $i-1$ takes ℓ more steps, giving a lower bound on the value of OPT.

Lemma 6 If $\Delta_i(t+1) = \Delta_i(t) + 1 > \Delta_{i-1}(t) \geq 0$, then, in OPT's schedule at time t, there exist $\Delta_i(t+1) - \Delta_{i-1}(t)$ packets with current priority at least $\Delta_i(t+1)+1$ that still need to be processed on router $i-1$.

Proof If $\Delta_i(t+1) = \Delta_i(t)+1$, GREEDY is idle on router i at time t, so GREEDY has processed all packets that were released on router i. Thus, before time t, OPT has processed at least $\Delta_i(t+1)$ length-2 packets that were released on router $i-1$ and that GREEDY has not processed on router $i-1$ before time t. Denote this set of packets by P and let p be a packet with minimum release time in P. Note that

$$t - r(p) \geq |P| \geq \Delta_i(t+1),$$

since OPT processes P on router $i-1$ between $r(p)$ and t.

Consider the sets of packets that GREEDY and OPT process on router $i-1$ before time t. Denote

these sets by G and O, respectively, and note that

$$|O| = |G| + \Delta_{i-1}(t). \tag{17}$$

Moreover, since $P \subseteq O \setminus G$,

$$|O \setminus G| \geq |P| \geq \Delta_i(t+1), \tag{18}$$

so

$$\begin{aligned} |G \setminus O| &= |O \setminus G| - \Delta_{i-1}(t), \text{ by (17) and removing } O \cap G \text{ from both } O \text{ and } G \\ &\geq \Delta_i(t+1) - \Delta_{i-1}(t), \text{ by (18).} \end{aligned}$$

Since all packets in P are released on router $i-1$, they are all available to Greedy on router $i-1$ as soon as they are released. Therefore, by the definition of Greedy and the fact that p has length 2, each packet in $G \setminus O$ is at least as old as p and has length 2 or is at least one time step older than p. This means that at time t, Opt has at least $\Delta_i(t+1) - \Delta_{i-1}(t)$ packets that each have a priority of at least $\Delta_i(t+1)+1$ and still need to be processed on router $i-1$. □

For an example of how Lemma 6 applies, see the lower bound proof as depicted in Figure 1. This input sequence is based on the blocks defined at the beginning of the proof of Theorem 2. Letting $i = 4$, we have

$$t = 17h + 2,$$

since $17h+2$ is the last time before finishing that Greedy is not processing a packet on router 4. Since Opt processes the last two packets of B_3 at times $t = 17h+2$ and $t+1 = 17h+3$ on router 3, the packets in the set P are the first $7h-2$ packets in B_3, and

$$\Delta_4(t+1) = \Delta_4(17h+3) = 7h-2.$$

The oldest packet p in P was released at time

$$r(p) = 10h + 3$$

(since they were all released then). By this time, Opt has completed all but the last packet of B_2, and Greedy has completed the first packet of B_2, so Opt was ahead of Greedy by

$$\Delta_3(r(p)) = 6h - 2$$

packets from B_2 on router 3.

At time $r(p)$ on router 3, Opt has not begun processing any of the h packets in A_3 on router 3, each of which has priority

$$\pi(A_3, t) = 2 + (17h+2) - (10h+1) - (2-1) = 7h+2$$

at time $t = 17h+2$. Thus, these are the packets in $G \setminus O$ from Lemma 6. According to the lemma, the last packet in A_3 which Opt processes on router 3 will have priority (and flow time) at least $\Delta_4(t+1) = 7h-2$, and there are at least

$$|A_3| = \Delta_4(t+1) - \Delta_3(t) = (7h-2) - (6h-2) = h$$

packets in A_3. In this example, the number of packets remaining for OPT on router 3 matches the lower bound in the lemma, and the priority of $7h+2$ is larger than the promised $\Delta_4(t+1)+1 = 7h-1$. As mentioned below in the proof of Lemma 7, at least one of the packets has flow time at least

$$2\Delta_4(t+1) - \Delta_3(t) = 2(7h-2) - (6h-2) = 8h-2$$

in OPT's schedule (its actual flow time is $8h+3$).

One problem with extending Lemma 6 to packets with length more than two, is that it is not clear that the delayed packets for OPT, those in $G \setminus O$, would only be on one router. Maybe in the worst case they are divided evenly over the previous routers. The length of the longest packet will probably enter into the competitive analysis in some way, though maybe only as an additive constant.

Lemma 7 For any router $i \geq 1$,

$$\max_{t \geq 0, j \leq i} \Delta_j(t) \leq \left(1 - \frac{1}{2^{i-1}}\right) \text{OPT}\,.$$

Proof We prove the lemma by induction on i. For $i = 1$, the inequality is true by Lemma 5, establishing the base case.

For the inductive step, consider router $j \geq 2$ and let $t+1$ be the first time that the maximum value of Δ_j is attained. Then, by Lemma 6, at time t there are $\Delta_j(t+1) - \Delta_{j-1}(t) > 0$ packets that OPT still needs to process on router $j-1$ and that already have a priority of at least $\Delta_j(t+1)+1$. The first to be processed may have flow time $\Delta_j(t+1)+1$, but the last one to be processed will accumulate a flow time of at least $2\Delta_j(t+1) - \Delta_{j-1}(t)$. Thus,

$$\begin{aligned}
\text{OPT} &\geq 2\Delta_j(t+1) - \Delta_{j-1}(t) \\
&\geq 2\Delta_j(t+1) - (1 - 2^{2-j})\,\text{OPT}, \text{ by the induction hypothesis} \\
&\geq 2\Delta_j(t+1) - (1 - 2^{2-i})\,\text{OPT}, \text{ since } j \leq i.
\end{aligned}$$

Hence,

$$(1 - 2^{1-i})\,\text{OPT} \geq \Delta_j(t+1).$$

This completes the proof. □

Theorem 8 If there are k routers and all packets have length 1 or 2, GREEDY is $\left(2 - \frac{1}{2^{k-1}}\right)$-competitive.

Proof Consider any router $i \geq 2$ and any packet p that is released at time $r(p)$ on router $i-1$ or i and that has router i as its last router. The priority of every packet that remains alive in GREEDY's schedule increases by 1 in every time step in which it is not processed. Hence, p is not delayed (on any router) by any packet that is released (on any router) at time $t' = r(p)+2$ or later since p has higher priority.

At time t', there are $g_{i-1}(t')$ packets alive (on router $i-2$ or $i-1$) that eventually need to be processed on router $i-1$, and

$$\begin{aligned}
g_{i-1}(t') &= a_{i-1}(t') + \Delta_{i-1}(t') \\
&\leq \text{OPT} + \left(1 - \frac{1}{2^{i-2}}\right) \text{OPT}, \text{ by Lemma 7} \\
&= \left(2 - \frac{1}{2^{i-2}}\right) \text{OPT}\,.
\end{aligned}$$

Similarly, at time t', there are

$$g_i(t') = a_i(t') + \Delta_i(t') \leq \left(2 - \frac{1}{2^{i-1}}\right) \textsc{Opt}$$

packets alive that need to be processed on router i.

Thus, if p is released at router i, its completion time is no later than $t' + g_i(t')$. If p is released at router $i-1$, it arrives at router i no later than at time $t' + g_{i-1}(t')$. Since no packet released at time t' or later can delay p, it cannot be delayed on router i until a time later than $t' + g_i(t')$. Therefore, the completion time of p is at most $t' + \max\{g_{i-1}(t') + 1, g_i(t')\}$.

Since $t' = r(p) + 2$, we conclude that p is completed no later than at time

$$\begin{aligned} r(p) + 2 + \max\{g_{i-1}(t') + 1, g_i(t')\} &< r(p) + 2 + \left(2 - \frac{1}{2^{i-1}}\right) \textsc{Opt} + 1 \\ &\leq r(p) + \left(2 - \frac{1}{2^{k-1}}\right) \textsc{Opt} + 3, \text{ since } i \leq k. \end{aligned}$$

Hence, the flow time of p is at most $\left(2 - \frac{1}{2^{k-1}}\right) \textsc{Opt} + 3$. □

4 General lower bounds

We start with a warm-up construction, before presenting the 4/3 lower bound for any randomized algorithm.

Theorem 9 Any randomized algorithm has a competitive ratio of at least 6/5 even with only $k = 2$ routers.

Proof Assume that Alg is a $(6/5 - \varepsilon)$-competitive algorithm for some $\varepsilon > 0$. We give a family of input sequences $\{I_h\}_{h \in \mathbb{N}}$. The sequence I_h starts with the following packets.

- There are $2h$ packets released at time 0 that need to be processed at router 1. We call these packets *short packets*.
- There are h packets released at time h that need to be processed at routers 1 and 2. We call these packets *long packets*.

Let $y \in [0, h]$ be the expected number of long packets that Alg processes on router 1 before time $2h$. The expected maximum flow time among the short packets must be at least $2h + y$. If no further packets are released, the optimal maximum flow time is $2h + 1$ by giving preference to the short packets at router 1. Thus, since Alg is 6/5-competitive, there must be a constant a such that

$$2h + y \leq \frac{6}{5} \cdot (2h + 1) + a,$$

or equivalently, by solving for y,

$$y \leq \frac{2h}{5} + b, \text{ where } b = a + \frac{6}{5}.$$

Thus, the expected number of packets that are left to be processed for Alg on router 2 at time $2h + 1$ is at least

$$h - y \geq \frac{3h}{5} - b.$$

If now $3h$ additional *jam packets* are released at time $2h+1$ that need to be processed by router 2, the expected number of packets that ALG still needs to process on router 2 will be at least

$$\frac{3h}{5} - b + 3h = \frac{18h}{5} - b$$

so the expected maximum flow time in ALG's schedule will be at least this number. The optimal maximum flow time for the entire instance, however, is $3h$ by giving preference to the long packets at router 1 (thus ensuring there is no conflict between the long packets and the jam packets). Hence, $\text{ALG}(I_h) \geq \frac{6}{5}\,\text{OPT}(I_h) - b$. For h large enough, this contradicts that ALG is $(6/5-\varepsilon)$-competitive. □

Next, we prove a lower bound of 4/3 for randomized algorithms, using the same idea as in the proof of Theorem 9, using "short" and "long" packets iteratively to build up how much behind OPT the randomized algorithm, ALG, is on a specific router. The proof also reuses the idea of restricting ALG to maintaining the goal competitive ratio during each iteration (since it is online), but allowing OPT to perform worse than ALG until the end where there are many "jam" packets.

We define an instance I to be (t, i, U, L)-*critical* if there exists an offline algorithm, OFF, such that the following hold:

(i) The cost of OFF on I is at most U.

(ii) At time t, OFF has no packets left to process on routers $i, i+1, \ldots, k$.

(iii) At time t, for every 4/3-competitive randomized algorithm ALG, the expected number of packets that ALG has left to process on router i is at least L.

First, we show the following lemma.

Lemma 10 For any 4/3-competitive algorithm, ALG, there exists a constant, b, such that for any nonnegative L and any positive even U, the existence of a (t, i, U, L)-critical instance, I, implies the existence of a $(t', i+1, 3U/2, L+U/6-b)$-critical instance, I', for some $t' \in \mathbb{N}$.

Proof To create the sequence I', the following packets are released in addition to those in I:

- At time t, U *short packets* that need to be processed at router i,
- At time $t+U/2$, $U/2$ *long packets* that need to be processed at routers i and $i+1$.

Let P denote the set consisting of the L packets waiting on router i at time t and the U packets released on router i at time t. Let ALG be a 4/3-competitive algorithm. Let $y \in [0, U/2]$ be the expected number of the long packets released at time $t+U/2$ that ALG processes on router i before time $t+U$. This means that the expected time at which ALG has processed all packets in P is at least $t+L+U+y$. Hence, the expected maximum flow time of the $L+U$ packets in P is at least $L+U+y$.

Note that based on OFF, we can define another offline algorithm, OFF$'$, such that $\text{OFF}'(I') \leq U+1$. This can be achieved by processing all of the short packets before the long ones. Therefore, since ALG is 4/3-competitive and OPT is at least as good as OFF$'$, there must exist a constant a, independent of U and L, such that

$$L + U + y \leq \frac{4}{3}(U+1) + a,$$

or equivalently, by solving for y,

$$y \leq \frac{U}{3} - L + b, \text{ where } b = a + \frac{4}{3}.$$

Thus, the expected number of packets that are left to be processed for ALG on router $i' = i + 1$ at time $t' = t + U + 1$ is at least

$$\frac{U}{2} - y \geq \frac{U}{6} + L - b.$$

Finally, by processing the long packets, starting as soon as they become available, before the rest of the short packets, we can define another offline algorithm OFF$''$ from OFF. Note that OFF$''$ can indeed guarantee a cost of at most $(3/2) \cdot U$ and having no packets left to be processed at routers $i', i' + 1, \ldots, k$ from time t'. □

Theorem 11 Any randomized algorithm has a competitive ratio of at least 4/3.

Proof Let $\varepsilon > 0$. Suppose ALG is $(4/3 - \varepsilon)$-competitive. We start with the empty instance $I^{(0)}$. Note that it is $(0, 0, k, 0)$-critical for any positive integer k. For a positive integer i to be set later as a function of ε and for a positive integer ℓ that we will later choose large, we set $k = 2^i \cdot \ell$, allowing us to apply Lemma 10 iteratively for i times to the $(0, 0, k, 0)$-critical instance $I^{(0)}$ (since it requires the third entry of the 4-tuple to be a multiple of 2).

We obtain an instance $I^{(i)}$ that is (t, i, U_i, L_i)-critical for some $t \in \mathbb{N}$ where

$$U_i = \left(\frac{3}{2}\right)^i \cdot k,$$

and, for some constant, b,

$$L_i = \sum_{j=0}^{i-1} \left(\frac{1}{6} \cdot U_j - b\right) \geq \left(\sum_{j=0}^{i-1} \left(\frac{2}{3}\right)^{i-j} \cdot \frac{1}{6}\right) \cdot U_i - bi = \left(\sum_{j=1}^{i} \left(\frac{2}{3}\right)^{j} \cdot \frac{1}{6}\right) \cdot U_i - bi$$
$$= \left(\frac{1}{3} - \frac{2^i}{3^{i+1}}\right) U_i - bi.$$

Choosing i and ℓ sufficiently large, we obtain

$$L_i > \left(\frac{1}{3} - \frac{\varepsilon}{4}\right) U_i - bi \geq \left(\frac{1}{3} - \frac{\varepsilon}{2}\right) U_i.$$

Thus, by Lemma 10, ALG as a 4/3-competitive algorithm has in expectation more than $(1/3 - \varepsilon/2) \cdot U_i$ packets left at time t to be processed on router i. Releasing U_i additional packets at time t to be processed on router i hence ensures that the expected cost of ALG is more than $(4/3 - \varepsilon/2) \cdot U_i$. On the other hand, by the definition of critical, OFF, and therefore also OPT, can handle $I^{(i)}$ and these additional packets still at cost U_i because it has no packets left to be processed on router i at time t. Since ℓ, and hence, U_i, can be chosen arbitrarily large, this is a contradiction to ALG being $(4/3 - \varepsilon)$-competitive. □

5 Open Problem

We conjecture that GREEDY has a constant competitive ratio (possibly 2) for packet routing, even when the packet length is unbounded.

References

[1] Udo Adamy, Christoph Ambühl, R. Sai Anand, and Thomas Erlebach. Call control in rings. *Algorithmica*, 47(3):217–238, 2007.

[2] W. Aiello, E. Kushilevitz, R. Ostrovsky, and A. Rosén. Adaptive packet routing for bursty adversarial traffic. *J. Comput. Syst. Sci.*, 60(3):482–509, 2000.

[3] Antonios Antoniadis, Neal Barcelo, Daniel Cole, Kyle Fox, Benjamin Moseley, Michael Nugent, and Kirk Pruhs. Packet forwarding algorithms in a line network. In *11th Latin American Theoretical Informatics Symposium (LATIN)*, volume 8392 of *Lecture Notes in Computer Science*, pages 610–621. Springer, 2014.

[4] B. Awerbuch, Y. Azar, and S. A. Plotkin. Throughput-competitive on-line routing. In *Proceedings of the 34th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 32–40, 1993.

[5] Nikhil Bansal, Tracy Kimbrel, and Maxim Sviridenko. Job shop scheduling with unit processing times. *Math. Oper. Res.*, 31(2):381–389, 2006.

[6] Martin Böhm, Nicole Megow, and Jens Schlöter. Throughput scheduling with equal additive laxity. In *International Conference on Algorithms and Complexity*, pages 130–143. Springer, 2021.

[7] A. Borodin, J. M. Kleinberg, P. Raghavan, M. Sudan, and D. P. Williamson. Adversarial queuing theory. *J. ACM*, 48(1):13–38, 2001.

[8] Bo Chen. Analysis of classes of heuristics for scheduling a two-stage flow shop with parallel machines at one stage. *Journal of the Operational Research Society*, 46(2):234–244, 1995.

[9] B. S. Chlebus, D. R. Kowalski, and M. A. Rokicki. Adversarial queuing on the multiple access channel. *ACM Transactions on Algorithms*, 8(1):5, 2012.

[10] Rathish Das and Hao Sun. Approximation hardness of resource scheduling. In *Proceedings of the 37th ACM Symposium on Parallelism in Algorithms and Architectures*, pages 46–61, 2025.

[11] Yann Disser, Max Klimm, and Elisabeth Lübbecke. Scheduling bidirectional traffic on a path. In Magnús M. Halldórsson, Kazuo Iwama, Naoki Kobayashi, and Bettina Speckmann, editors, *Automata, Languages, and Programming - 42nd International Colloquium, ICALP 2015, Kyoto, Japan, July 6-10, 2015, Proceedings, Part I*, volume 9134 of *Lecture Notes in Computer Science*, pages 406–418. Springer, 2015.

[12] T. Erlebach and K. Jansen. Call scheduling in trees, rings and meshes. In *Proceedings of the Thirtieth Hawaii International Conference on System Sciences*, volume 1, pages 221–222 vol.1, 1997.

[13] Jessen T. Havill. Online packet routing on linear arrays and rings. In Fernando Orejas, Paul G. Spirakis, and Jan van Leeuwen, editors, *Automata, Languages and Programming, 28th International Colloquium, ICALP 2001, Crete, Greece, July 8-12, 2001, Proceedings*, volume 2076 of *Lecture Notes in Computer Science*, pages 773–784. Springer, 2001.

[14] Dylan Hyatt-Denesik, Mirmahdi Rahgoshay, and Mohammad R Salavatipour. Approximations for throughput maximization. *Algorithmica*, 86(5):1545–1577, 2024.

[15] Sungjin Im and Benjamin Moseley. Scheduling in bandwidth constrained tree networks. In Guy E. Blelloch and Kunal Agrawal, editors, *Proceedings of the 27th ACM on Symposium on Parallelism in Algorithms and Architectures, SPAA 2015, Portland, OR, USA, June 13-15, 2015*, pages 171–180. ACM, 2015.

[16] 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.

[17] F. T. Leighton, B. M. Maggs, and S. Rao. Packet routing and job-shop scheduling in o(congestion + dilation) steps. *Combinatorica*, 14(2):167–186, 1994.

[18] Qingsong Liu and Zhixuan Fang. Online task scheduling and termination with throughput constraint. *IEEE/ACM Transactions on Networking*, 32(6):4629–4643, 2024.

[19] R. Ostrovsky and Y. Rabani. Universal o(congestion + dilation + log1+epsilon n) local control packet switching algorithms. In *Proceedings of the 29th Annual ACM Symposium on Theory of Computing (STOC)*, pages 644–653, 1997.

[20] Boaz Patt-Shamir and Will Rosenbaum. Space-optimal packet routing on trees. In *2019 IEEE Conference on Computer Communications, INFOCOM 2019, Paris, France, April 29 - May 2, 2019*, pages 1036–1044. IEEE, 2019.

[21] A. Soukhal, A. Oulamara, and P. Martineau. Complexity of flow shop scheduling problems with transportation constraints. *European Journal of Operational Research*, 161(1):32–41, 2005. IEPM: Focus on Scheduling.

[22] Pavel Veselý, Marek Chrobak, Łukasz Jeż, and Jiří Sgall. *A ϕ-Competitive Algorithm for Scheduling Packets with Deadlines*, pages 123–142.

[23] Hongjun Wei and Jinjiang Yuan. Two-machine flow-shop scheduling with equal processing time on the second machine for minimizing total weighted completion time. *Operations Research Letters*, 47(1):41–46, 2019.