Online Packet Scheduling under Adversarial Jamming[*]

Tomasz Jurdzinski[†] Dariusz R. Kowalski[‡] Krzysztof Lorys[†]

Abstract

We consider the problem of scheduling packets of different lengths via a directed communication link prone to jamming errors. Dynamic packet arrivals and errors are modelled by an adversary. We focus on estimating relative throughput of online scheduling algorithms, that is, the ratio between the throughputs achieved by the algorithm and the best scheduling for the same arrival and error patterns. This framework allows more accurate analysis of performance of online scheduling algorithms, even in worst-case arrival and error scenarios. We design an online algorithm for scheduling packets of arbitrary lengths, achieving optimal relative throughput in $(1/3, 1/2]$ (the exact value depends on packet lengths). In other words, for any arrival and jamming patterns, our solution gives throughput which is no more than c times worse than the best possible scheduling for these patters, where $c \in [2, 3)$ is the inverse of relative throughput. Another algorithm we design makes use of additional resources in order to achieve relative throughput 1, that is, it achieves at least as high throughput as the best schedule without such resources, for any arrival and jamming patterns. More precisely, we show that if the algorithm can run with double speed, i.e., with twice higher frequency, then its relative throughput is 1. This demonstrates that throughput of the best online scheduling algorithms scales well with resource augmentation.

Keywords: Packet scheduling, Dynamic packet arrivals, Adversarial jamming, Online algorithms, Relative throughput, Resource augmentation.

1 Introduction

Motivation. Achieving high-level reliability in packet scheduling has recently become more and more important due to substantial increase of the scale of networks

[†]Institute of Computer Science, University of Wrocław, Poland.

[‡]Department of Computer Science, University of Liverpool, Liverpool L69 3BX, UK.

[*]This work was supported by the Polish National Science Centre grant DEC-2012/06/M/ST6/00459.

and higher fault-tolerant demands of many incoming applications. In the era of Internet of Things and nano-devices, it will no longer be possible to attend devices physically, and therefore the designed protocols must be stable and robust no matter of failure pattern. Imagine the problem of thousands of malfunctioning nano-capsules with overflown buffers that need to be somehow removed from the human body, or the consequences of lack of communication between AVs with humans onboard or medical devices incorporated into patients bodies, even if such case might happen with probability less than 1%.

Our Approach. This paper studies a fundamental problem of online packet scheduling via *unreliable* link (also called a channel), when transmitted packets may be interrupted by *unrestricted* jamming errors. This problem was recently introduced in [4] and analyzed for two different packet lengths. Packets arrive dynamically to one end of the link, called a sender, and need to be transmitted in full, i.e., without any in-between jamming error, to the other end (called a receiver). Jamming errors are immediately discovered by the sender. We analyze all possible scenarios, including worst case ones, which we model as a conceptually adversary who controls both packet arrivals and channel jamming. The adversary is unrestricted, in the sense that she may generate *any* arrival and error pattern. The main objective of the online scheduling protocol is to achieve as high throughput as possible under current scenario. In particular, we consider the measure called relative throughput, which is a long-term form of competitive ratio between the throughput achieved by the online algorithm and the one reached by optimum offline scheduling solution (i.e., under the knowledge of adversarial arrivals and errors).

Our Contribution. We design a deterministic online scheduling algorithm achieving optimal relative throughput for an arbitrary number k of packet lengths $\ell_{\min} = \ell_1 < \ell_2 < \ldots < \ell_k = \ell_{\max}$ (Section 3). We first show a simpler version of the algorithm, for the case when packet lengths are pairwise divisible (any larger is divisible by any smaller), in order to demonstrate high-level ideas and analysis leading to related throughput $1/2$. We then extend the protocol so that it does not need to rely on such limitation about divisibility, and achieves the relative throughput $\min_{1\le j<i\le k}\left\{\frac{\lfloor\rho_{i,j}\rfloor}{\lfloor\rho_{i,j}\rfloor+\rho_{i,j}}\right\}$, where $\rho_{i,j} = \ell_i/\ell_j$ is the ratio between the i-th and the j-th packet length. Note that this general formula for relative throughput is in the range $(1/3, 1/2]$, and it reaches $1/2$ if and only if the pairwise divisibility condition holds.

Unfortunately, the designed protocol does not achieve relative throughput 1 if the speed-up 2 is applied (it can be easily checked that the relative throughput is at most $2/3$ in such case), which implies that it is not well-scalable with resource augmentation.[1] Therefore we design another deterministic online protocol to op-

[1]Note that the considered speed-up 2 is chosen because we claim linear scalability of relative

timize relative throughput for speedup 2 (Section 4). It is a generalisation of the preamble protocols proposed in [4] and [5] in the case of two packet lengths.

More details can be found in the full draft of the paper [7].

Previous and related work. Packet scheduling [9] is one of the most fundamental problems in computer networks. A realistic approach involves *online* scheduling [6, 11], and therefore a *competitive analysis* [1, 14] is often used to evaluate the performance of proposed solutions. Online scheduling was considered in a number of models; for more information the reader is referred to [10] and [11].

The framework considered in this work was recently introduced in [4]. The authors showed that general offline version of this problem, in which the scheduling algorithm knows a priori when errors will occur, is NP-hard. They also considered algorithms and upper limitations for relative throughput in case of *two* packet lengths. In particular, they proved that relative throughput of *any* online scheduling protocol cannot be bigger than $\overline{\rho}/(\rho + \overline{\rho})$, where ρ is the ratio between the bigger and the smaller packet length and $\overline{\rho} = \lfloor \rho \rfloor$. (Note that the upper bound becomes $1/2$ if the bigger packet length is a multiplicity of the smaller packet length.) This upper bound can be achieved by a protocol scheduling a specific preamble of shorter packets followed by the Longest_First rule after every error, but cannot be reached by simpler protocols such as Longest_First itself or Shortest_First (in fact, the relative throughputs of the latter protocols are far worse than $1/2$: 0 and $1/(1+\rho)$, respectively, and thus they are not very reliable). Therefore, it remained open whether there is an online scheduling protocol reaching the relative throughput of (roughly) $1/2$ for *arbitrary* number of packet lengths; we answer this question in affirmative in this work. Moreover, as also shown in [4], randomization does not help, which motivates our study of deterministic algorithms.

In [5], the authors studied buffer sizes of online scheduling protocols on error-prone channel. Unlike the relative throughput measure, in order to be positively competitive with the best scheduling algorithms with respect to the buffer sizes, additional resources need to be given to the online protocol, i.e., speed-up (higher frequency). This form of resource allocation appeared to be efficient: for some speed-up smaller than 2 there is a deterministic online scheduling algorithm having roughly the same queue sizes as any other scheduling algorithm running without speed-up. That work motivated us to consider resource augmentation technique, in the form of using some speed-up (higher frequency), to reach at least the same throughput as the best scheduler without speed-up for *every* execution.

Wireless packet scheduling was also considered in models with physical constraints included, such as radio networks or SINR. Anantharamu et al. [2] consid-

throughput with the increase of speed-up, that is, starting from level $1/2$ with no speed-up we expect the relative throughput to reach value 1 for speed-up 2.

ered packet scheduling on a multiple access channel with signal interference, under a restricted adversarial patterns of packet arrivals and channel jamming. Kesselheim [8] considered packet scheduling problem in the SINR model, for both adversarial and stochastic arrivals, but with no errors. Both papers studied stronger objective measure: maximum time from packet arrival to successful delivery. In this line of research, the most relevant direction was taken by Richa et al. [13] who analyzed competitive throughput of randomized scheduling protocols on multiple access channels with signal interference against *adaptive, but still restricted, adversarial jamming*. Therefore, the results obtained in this line of work cannot be directly comparable with ours, mainly because of assuming restricted arrival and jamming patterns.

Andrews and Zhang [3] studied buffer stability (i.e., bounded buffers property) of online packet scheduling on a wireless channel, where both the channel conditions and the data arrivals are controlled by an adversary. They also assumed bounded adversary, as otherwise stability could not be reached.

Our framework could also be applied to other types of channel errors, as long as the feedback is immediately delivered to the sender, e.g., to the emerging Continuous Error Detection (CED) framework [12] that uses arithmetic coding to provide continuous error detection.

2 Model

We consider a uni-directional point-to-point link in which one end point, called a *sender*, transmits packets to the other end point, called a *receiver*. The sender is equipped with unlimited buffer, in which the arriving packets are queued. Packets may be of different lengths, and may arrive at any time; we assume that time is continuous, and scheduling algorithm have access to packets as soon as they arrive. There are $k \geq 2$ different packet lengths, denoted by $\ell_{\min} = \ell_1 < \ell_2 < \ldots < \ell_k = \ell_{\max}$. For simplicity, we will use the names "ℓ_i-packets" and "packets ℓ_i" for packets of length ℓ_i, for any $1 \leq i \leq k$. For clarity of presentation, we assume in some parts of the paper that ℓ_i/ℓ_j is an integer for any $1 \leq j < i \leq k$ (so called pairwise divisibility property). We denote $\rho = \ell_{\max}/\ell_{\min}$. We assume that all packets are transmitted at the same bit rate, hence the transmission time is proportional to the packet's length. The link is prone to jamming errors, that is, transmitted packets might be corrupted at any time point.

Arrival models. We consider adversarial packet arrivals: the packets' arrival time and length are governed by an adversary. We define an adversarial arrival pattern as a collection of packet arrivals caused by the adversary.

Link jamming errors. We consider adversarial model of jamming errors, in which the adversary decides at which time to cause a jamming error on the link. The error at time t implies that any packet being transmitted at time t is broken, and the information about it is immediately delivered to the sender so that it breaks the current transmission and could schedule another packet (or re-schedule the one that was just broken). A corrupted packet transmission is unsuccessful, in the sense that it is not received by the receiver and it needs to be retransmitted in full (not necessarily right after the error — scheduling algorithm may decide to postpone it and transmit another packet instead); otherwise it is understood as not (successfully) transmitted. We assume that scheduling algorithms do not voluntarily stop transmitting packets before the end of the transmission, unless they get feedback about jamming error. An adversarial error pattern is defined as a collection of error events on the link caused by the adversary.

Adversarial models are typically used to argue about the algorithm's behavior in any possible scenario, in particular, in the worst-case ones.

Efficiency metric: *Relative throughput.* We would like to measure throughput of the communication link. However, due to adversarial errors, the real link capacity may vary in time, and moreover, due to adversarial packet arrivals, the stream of packets may not be regular or saturated. Therefore, following [4], we pursue a long-term competitive analysis. Let $\mathcal{A}$ be an arrival pattern and $\mathcal{E}$ an error pattern. For a given deterministic algorithm ALG, let $L_{ALG}(\mathcal{A}, \mathcal{E}, t)$ be the total length of all the successfully transferred (i.e., non-corrupted) packets by time t under arrival pattern $\mathcal{A}$ and error pattern $\mathcal{E}$. Let OPT be the offline optimal algorithm that knows the exact arrival and error patterns, as well as the online algorithm, before the start of the execution. We assume that OPT devises an optimal schedule that minimises the asymptotic ratio (i.e., with time growing to infinity) between the total length of packet transmitted by the online algorithm and the total length of packet transmitted by itself.

We require that any pair of patterns $\mathcal{A}, \mathcal{E}$ occurring in an execution must allow non-trivial communication, i.e., the value of $L_{OPT}(\mathcal{A}, \mathcal{E}, t)$ in the execution is unbounded with t going to infinity.

For arrival pattern $\mathcal{A}$, adversarial error pattern $\mathcal{E}$ and time t, we define the *relative throughput $T_{ALG}(\mathcal{A}, \mathcal{E}, t)$ of a deterministic algorithm ALG by time t* as:

$$T_{ALG}(\mathcal{A}, \mathcal{E}, t) = \frac{L_{ALG}(\mathcal{A}, \mathcal{E}, t)}{L_{OPT}(\mathcal{A}, \mathcal{E}, t)} .$$

For completeness, $T_{ALG}(\mathcal{A}, \mathcal{E}, t)$ equals 1 if $L_{ALG}(\mathcal{A}, \mathcal{E}, t) = L_{OPT}(\mathcal{A}, \mathcal{E}, t) = 0$.

We define the *relative throughput* of ALG in the adversarial arrival model as:

$$T_{ALG} = \inf_{\mathcal{A}, \mathcal{E}} \lim_{t \to \infty} T_{ALG}(\mathcal{A}, \mathcal{E}, t) . \quad (1)$$

In the analysis of lower and upper bound on relative throughput, we usually focus on comparison of the number of successful transmissions of packets, weighted by packet lengths, for periods after *sufficiently large* time t. This is because the performances of online and optimal algorithms in a fixed prefix of time are negligible from perspective of the definition of relative throughput given in Equation (1).

Resource augmentation — speed-up. In the second part of the paper, in Section 4, we consider resource augmentation technique. This technique was recently applied to fault-tolerant scheduling in [5] in the context of buffer stability metric. In particular, we compare the throughput of a given online algorithm under the assumption that this algorithm is run with a certain speed-up $s > 1$, with the throughput of the best scheduling algorithm run without any speed-up. From technical perspective, computing of the relative throughput under speed-up $s > 1$ follows the same definitions as given above, with the only difference that the value of $L_{ALG}(\mathcal{A}, \mathcal{E}, t)$ is calculated under assumption that ALG transmits packets s times faster.

3 Packet Scheduling for k packet lengths

In this section we present online algorithm, which is optimal for any number of packet lengths $k \geq 2$. First, for the ease of presentation, we present algorithm Greedy under assumption that $\ell_i/\ell_{i-1} \in \mathbb{N}$ for $1 < i < k$. Later, in Section 3.1, we show how to remove this assumption by modifying algorithm Greedy; the resulted algorithm is called MGreedy.

The main idea behind our algorithms is to keep transmitting as many short packets as possible (shortest-first strategy), subject to some balancing constraints. Observe that it is difficult for any offline algorithm OFF to get advantage over any online algorithm ALG when ALG sends small packets. Thus, preference for small packets ensures that ALG can be competitive against OFF, as long as it has short packets. However, if OFF transmits large packets during transmission of small packets by ALG, it can afterwards transmit small packets when ALG does not have any of them in its queue. Simultaneously, when OFF is transmitting small packets, ADV can generate errors preventing ALG from successful transmission of large packets. Despite this disadvantage of a greedy approach, we show that an appropriate implementation of this strategy, using some balancing constraints, provides an optimal solution with respect to relative throughput, and thus against any optimal way of scheduling under occurring arrival and failure patterns.

Our specific modification of the greedy shortest-first strategy is based on sending packets in groups, which altogether balance the length of the next larger packet. We explain it in detail first for two types of packet lengths: $\ell_{\min}$ and $\ell_{\max}$. If there are at least $\rho = \ell_{\max}/\ell_{\min}$ small packets in the queue, the algorithm builds a *group* which consists of ρ of them and keeps sending them until all of them are transmitted successfully. If there are less than ρ small packets in the queue at the moment when a transmission of a group is finished, a large packet is transmitted. However, whenever there are at least ρ small packets, the group of small packets is formed, independently of the fact whether a transmission of a large packet(s) is successful or not. This idea is then recursively applied for the case when there are $k > 2$ types of packets. A pseudo-code of our greedy algorithm is presented as Algorithm 1, with its recursive subroutine given as Algorithm 2.

Algorithm 1 Greedy

```
1: loop
2:     while $\sum_{i=1}^{k} \ell_i n_i < \ell_k$ do Stay idle
3:     Transmit-group($k$)
```

Algorithm 2 Transmit-group(j)

```
1: loop
2:     if $\sum_{i=1}^{j-1} \ell_i n_i \geq \ell_j$ then
3:         for $a = 1$ to $\ell_j/\ell_{j-1}$ do Transmit-group($j-1$)
           return
4:     Transmit $\ell_j$; If the transmission is successful: return
```

In the pseudo-codes, n_i denotes the number of packets ℓ_i which are currently (at the moment) waiting in the queue for transmission.

Performance analysis of algorithm Greedy.

For the sake of analysis of algorithm Greedy, we introduce some new notations. First, let us assume that an arrival pattern and an injection pattern are chosen arbitrarily and are fixed, so we could omit them from formulas in the further analysis. For an algorithm A, let $q_A(i,t)$ denote the sum of lengths of ℓ_i-packets in the queue of A at the moment t. That is, $q_A(i,t) = n_i \cdot \ell_i$ for a fixed time t. Moreover, let $q_A(< i,t) = \sum_{j<i} q_A(j,t)$ and we define $q_A(\leq i,t)$ analogously. Let $L_A(i,t)$ denote the length of packets ℓ_i successfully transmitted by time t. For a time period $\tau = [t_1, t_2]$, let $L_A(i,\tau) = L_A(i,t_2) - L_A(i,t_1)$. That is, $L_A(i,\tau)$ denotes the number of ℓ_i-packets successfully transmitted in the interval τ. The notions $L_A(< i,t)$, $L_A(\leq i,t)$, $L_A(< i,\tau)$, and $L_A(\leq i,\tau)$ for time t and time interval τ are defined analogously to $q_A(< i,t)$, $q_A(\leq i,t)$, $q_A(< i,\tau)$ and $q_A(\leq i,\tau)$. We also use the above introduced notations without the first argument, i.e., $q_A(t)$, $q_A(\tau)$, $L_A(t)$, and $L_A(\tau)$, which are shorthands for $q_A(\leq k,t)$, $q_A(\leq k,\tau)$, $L_A(\leq k,t)$ and $L_A(\leq k,\tau)$, respectively.

An algorithm A is *busy* at time t if it is transmitting a packet at t, it has just finished a successful transmission, or its transmission is jammed by an error at t. Otherwise A is *idle* at t.

Our goal is to compare progress in sending packets of our algorithm Greedy and an algorithm achieving the optimal throughput, denoted as OFF. We say that an algorithm A is *m-busy* in a time period $\tau = [t_1, t_2]$ if the following conditions are satisfied:

1. A is busy at each time $t \in \tau$;
2. A does not transmit packets ℓ_i for $i > m$ during τ;
3. $q_A(i, t_1) \geq q_{OFF}(i, t_1)$ for each $i \in [m]$. (That is, at time t_1 A has no less packets of length ℓ_i in its queue than OFF, for each $i \leq m$.)

Now, we prove technical results regarding periods in which Greedy is m-busy for some $m \in [k]$. These lemmas eventually lead to the proof of the fact that relative throughput of Greedy is $1/2$ (provided $\ell_i/\ell_{i-1} \in \mathbb{N}$ for $i \in [2, k]$), which is optimal. First, we make an observation that, if Greedy does not use packets longer than ℓ_m for $m \in [k]$, then the total length of packets transmitted by Greedy is at least as large as the total length of packets of length at least ℓ_m transmitted by OFF.

Lemma 1. *Assume that Greedy is m-busy in a time period τ, $m \leq k$. Then, $L_{Greedy}(\tau) \geq L_{OFF}(\geq m, \tau) - \ell_k$.*

Proof. Consider any packet ℓ_i for $i \geq m$ successfully transmitted by OFF in the period $[t, t + \ell_i] \subseteq \tau$. According to the assumptions, Greedy does not send any packets ℓ_j for $j > m$ in τ and it is busy at each time $t' \in \tau$. Therefore, Greedy finishes transmissions of ℓ_i/ℓ_m groups of packets of length ℓ_m in the period $[t, t + \ell_i]$. Thus, by assigning each group G of packets of length ℓ_m transmitted by Greedy to a packet transmitted by OFF at the moment when the last packet of G is finished, we obtain the result of the lemma. The "$-\ell_k$" reduction in the formula is needed in order to take into account the packet (if any) which OFF started transmitting before τ and finished in τ. $\square$

Next, we formulate a relationship between the length of packets transmitted by Greedy and OFF up to the moment when Greedy is transmitting the longest packet used by itself in the computation.

Lemma 2. *Assume that Greedy is m-busy in a time period $\tau = [t_1, t_2]$, $m \leq k$. Let $t \in \tau$ be any time at which Greedy starts transmitting ℓ_m. Then,*

$$2L_{Greedy}([t_1, t]) \geq L_{OFF}([t_1, t]) + q_{OFF}(< m, t) - \ell_m - \ell_k.$$

Proof. The idea is that each packet p successfully transmitted by Greedy is associated to:

(a) transmission of p by OFF;

(b) transmission of a packet ℓ_i for $i \geq m$ by OFF which lasted while the group G of length ℓ_i containing p was finished by Greedy.

This association guarantees that:

- each packet ℓ_i, for $i > m$, transmitted by OFF has an association (of type (b)) with a group of packets of lengths ℓ_i transmitted by Greedy;
- each packet ℓ_i, for $i \leq m$, transmitted by OFF has an association (of type (a)) with its transmission by Greedy, provided Greedy transmitted this packet successfully as well.

On the other hand, each packet p successfully transmitted by Greedy corresponds to successful transmissions of OFF with length at most twice the length of p. However, packets which are successfully transmitted by Greedy in $[t_1, t]$ and are not transmitted by OFF in $[t_1, t]$ "pay" for transmissions of OFF only once, i.e., are associated to a packet ℓ_i, for $i > m$, transmitted by OFF but not to transmissions of themselves by OFF. As Greedy tries transmitting ℓ_m at time t only in the case when $q_{Greedy}(< m, t) < \ell_m$, the claimed result holds.

The "$-\ell_k$" reduction in the formula is needed in order to take into account the packet which OFF started transmitting before τ and finished in τ. □

Using previous lemmas, we prove by induction a relationship between $L_{Greedy}(\tau)$ and $L_{OFF}(\tau)$ for periods τ which are m-busy for Greedy, where $m \in [k]$.

Lemma 3. *Assume that Greedy is m-busy in a time period τ, for $m \leq k$. Then,*

$$2L_{Greedy}(\tau) \geq L_{OFF}(\tau) - f_m$$

where f_m satisfies the relationships $f_1 = \ell_k$ and $f_{i+1} = f_i + 3\ell_{i+1} + \ell_i + \ell_k$ for $i \in [1, k-1]$.

Proof. The proof goes by induction with respect to m. For $m = 1$ the result is an immediate consequence of Lemma 1.

For the inductive step, assume that the result holds for some $m < k$. We will show the correctness of the result for the case when the longest packet sent by Greedy in τ is ℓ_{m+1}. We split τ into three subintervals:

- τ_1 from the beginning of τ to time t at which Greedy starts (an attempt to) transmitting ℓ_{m+1} for the last time during τ;

- τ_2 from t to $t' \in \tau$ such that either Greedy finishes a successful transmission of ℓ_{m+1} at t' or it gives up scheduling packets ℓ_{m+1} at t' (since it has enough shorter packets in the queue at t' to cover the length ℓ_{m+1});

- τ_3 from t' to the end of τ.

Lemma 2 implies that

$$2L_{Greedy}(\tau_1) \geq L_{OFF}(\tau_1) + q_{OFF}(< m+1, t) - \ell_{m+1} - \ell_k,$$

where t is the moment when τ_1 ends. Consider OFF' which acts in τ_2 and τ_3 as OFF, however: it starts τ_2 without packets of length ℓ_i for each $i < m+1$, and it stays idle each time OFF is transmitting a packet which was in its queue at the beginning of τ_2 and therefore it was not in the queue of OFF'.

Note that Greedy finishes an attempt to transmit a packet ℓ_{m+1} not later than at the moment when it successfully transmits packet ℓ_{m+1} or new packets shorter than ℓ_{m+1} of overall length at least ℓ_{m+1} are inserted in the queue and error occurs. Observe also that OFF' starts time interval τ_2 with an empty queue and it cannot finish transmitting a packet ℓ_i, for $i > m+1$, in τ_2 (if a time period of length ℓ_{m+1} without error occurs, τ_2 is finished by its definition). Thus,

$$L_{Greedy}(\tau_2) \geq L_{OFF'}(\tau_2) - 2\ell_{m+1}\ ,$$

since new packets inserted during τ_2 and transmitted by OFF' have length $< \ell_{m+1}$:

- packets of length smaller than ℓ_{m+1} are inserted until the beginning of the last attempt to send ℓ_{m+1} by Greedy in τ_2;

- packets of length at most ℓ_{m+1} are successfully transmitted by OFF' during the last attempt to send ℓ_{m+1} by Greedy, since this attempt takes a time period of length at most ℓ_{m+1}.

At the beginning of τ_3, it holds that $q_{Greedy}(i, t') \geq q_{OFF'}(i, t')$ for $i \leq m$, since Greedy attempted only transmitting one copy of ℓ_{m+1} during τ_2.

Therefore, the inductive hypothesis apply to the period τ_3 for the largest packet ℓ_m and OFF' in place of OFF:

$$2L_{Greedy}(\tau_3) \geq L_{OFF'}(\tau_3) - f_m\ .$$

Next, recall that OFF' differs from OFF only such that it does not transmit packets of lengths smaller than ℓ_{m+1}, which appear in the queue of OFF at time t, of total length $q_{OFF'}(< m+1, t)$. Thus,

$$L_{OFF'}(\tau_2 \cup \tau_3) \geq L_{OFF}(\tau_2 \cup \tau_3) - q_{OFF}(< m+1, t)\ .$$

All these inequalities summed up and combined with the fact that $L_{Greedy}(\tau) = L_{Greedy}(\tau_1) + L_{Greedy}(\tau_2 \cup \tau_3)$ yield:

$$\begin{aligned} 2L_{Greedy}(\tau) &\geq L_{OFF}(\tau_1) + (q_{OFF}(< m+1, t) - \ell_{m+1} \\ &\quad -\ell_k) + L_{OFF'}(\tau_2 \cup \tau_3) - f_m - 2\ell_{m+1} \\ &\geq L_{OFF}(\tau) - f_m - 3\ell_{m+1} - \ell_m - \ell_k \\ &\geq L_{OFF}(\tau) - f_{m+1} \ . \end{aligned}$$

□

Theorem 1. *The relative throughput of Greedy is equal to* $1/2$*, provided* $l_i/l_{i-1} \in \mathbb{N}$ *for each* $i \in [2, k]$.

Proof. Lemma 3 implies that the relative throughput gets arbitrarily close to $1/2$ on sufficiently long time intervals in which Greedy is busy and OFF starts with the queue containing smaller or equal number of packets of each size. On the other hand, Greedy gets idle only in the case when its queue is almost empty, i.e., contains packets of length $< \ell_k$, which means that its relative throughput is arbitrarily close to 1 in such a moment t, provided t is large enough.

The theorem holds by combining these two observations:

- It holds at the moments when Greedy is idle.
- We can assume that OFF has empty queues at the moments when Greedy starts being idle — this assumption does not improve relative throughput of Greedy (we allow OFF to transmit all packets from its queue immediately in a period of length 0). Thus, the assumption of Lemma 3 is satisfied when Greedy starts transmitting after being idle. Therefore, the relative throughput of Greedy gets arbitrary close to $\frac{1}{2}$ in sufficiently large periods in which Greedy is not idle.

□

Corollary 1. *The algorithm Greedy achieves optimal relative throughput for packets' lengths* $\ell_1 < \ldots < \ell_k$ *such that* $\ell_i/\ell_{i-1} \in \mathbb{N}$ *for each* $i \in [2, k]$.

Proof. It is shown in [4] that relative throughput of any online algorithm for two types of packets is at most

$$\frac{\lceil \ell_2/\ell_1 \rceil}{\lceil \ell_2/\ell_1 \rceil + \ell_2/\ell_1}$$

which is equal to $\frac{1}{2}$ when $\ell_2/\ell_1 \in \mathbb{N}$. As an adversary can decide to schedule merely two types of packets among available k types, Theorem 1 implies optimality of relative throughput of Greedy. □

3.1 Arbitrary lengths of packets

In this section we discuss an application of the ideas behind the algorithm Greedy to the general case, i.e., when the condition $\ell_i/\ell_{i-1} \in \mathbb{N}$ is not satisfied. Let $\rho_{i,j} = \ell_i/\ell_j$. A natural generalization of Greedy is that, instead of ρ_i groups of packets of length ℓ_{i-1} on the i-th level of recursion, we choose $\lfloor \rho_{i,i-1} \rfloor$ groups of packets of length (as close as possible to) ℓ_{i-1} in order to "cover" ℓ_i. If the length of a group of packets on the i-th level of recursion is not larger than ℓ_{i+1}, we can apply the ideas of "covering" packets transmitted by OFF using groups of packets transmitted by Greedy. If $k = 2$, this approach gives an algorithm with relative throughput $\frac{\lfloor \rho_{2,1} \rfloor}{\lfloor \rho_{2,1} \rfloor + \rho_{2,1}}$, which is optimal due to [4]. This naturally generalizes to the following result.

Theorem 2. *The relative throughput of any online scheduling algorithm is at most*

$$\min_{1 \le j < i \le k} \left\{ \frac{\lfloor \rho_{i,j} \rfloor}{\lfloor \rho_{i,j} \rfloor + \rho_{i,j}} \right\} .$$

Proof. This result easily follows from Theorem 1 in [4]. Indeed, if the adversary schedules merely packets ℓ_i and ℓ_j, for i, j minimizing the expression $\frac{\lfloor \rho_{i,j} \rfloor}{\lfloor \rho_{i,j} \rfloor + \rho_{i,j}}$, the strategy described in the proof of Theorem 1 in [4] gives our result. $\square$

However, for $k > 2$, the additional advantage of OFF over Greedy following from rounding on various levels of recursion can accumulate. In order to limit this effect, instead of transmitting $\lfloor \ell_i/\ell_{i-1} \rfloor$ groups of packets on the level $i-1$, we keep sending groups on the level $i-1$ as long as the sum of lengths of packets from the transmitted groups is not larger than $\ell_i - \ell_{i-1}$. This gives the following technical result. (For simplifying the arguments in the remaining part of the analysis, let us denote $\rho_{i,i-1} = \ell_i/\ell_{i-1}$ by simply ρ_i, for $i \in [2, k]$.)

Lemma 4. *Consider such a modification of Greedy that Transmit-group(j) keeps calling Transmit-group$(j-1)$, for $j > 1$, as long as the total length of transmitted packets in the current execution of Transmit-group(j) is at most $\ell_j - \ell_{j-1}$. The relative throughput of this algorithm is at least* $\min_{i \in [2,k]} \left\{ \frac{\rho_i - 1}{2\rho_i - 1} \right\}$.

Proof. In Lemmas 1, 2 and 3, we repeatedly use an argument that, if Greedy does not use packets of length ℓ_i for $i > m$, then each such packet transmitted by OFF corresponds to a group of (shorter) packets transmitted by Greedy of total length ℓ_i. This observation can be preserved for the modified Greedy algorithm with a relaxation that a packet ℓ_i transmitted by OFF corresponds to a group of packets transmitted by Greedy of length at least $\ell_i - \ell_{i-1}$. This relaxation translates inequalities from Lemmas 1, 2 and 3 to:

$$\begin{array}{rcl} \frac{\rho_m}{\rho_m - 1} \cdot L_{Greedy}(\tau) & \geq & L_{OFF}(\geq m, \tau) - \ell_k \\ (1 + \frac{\rho_m}{\rho_m - 1}) \cdot L_{Greedy}([t_1, t]) & \geq & L_{OFF}([t_1, t]) + \\ & & q_{OFF}(< m, t) - \ell_m - \ell_k \\ (1 + \frac{\rho_m}{\rho_m - 1}) \cdot L_{Greedy}(\tau) & \geq & L_{OFF}(\tau) - f_m \end{array}$$

If we apply the above inequalities instead of those from Lemmas 1, 2 and 3 in the proof of Theorem 1, we obtain the result claimed here. □

However, as a group of packets transmitted by Greedy "covering" ℓ_i transmitted by OFF may contain packets of various lengths, the relative throughput of the solution from Lemma 4 is difficult to compare with the upper bound from Theorem 2. In order to tackle this issue, we introduce yet another modification to the algorithm.

The main goal of this modification is to ensure that Greedy is transmitting packets of the same length for long periods of time and it changes to other length only if it is necessary. An execution of the algorithm is split into *stages*. In a stage, packets of total length (close to) $ck\ell_k$ are transmitted, where $c \in \mathbb{N}$ is a fixed large constant. At the beginning of a stage, the set C of candidates is determined as $C = \{i \,|\, n_i \ell_i \geq ck\ell_k\}$. Then, the *interesting* length $\ell_{i^\star}$ is set for parameter $i^\star = \min(C)$, and the algorithm starts transmitting packets $l_{i^\star}$. After each transmission, successful or not, the interesting length $i^\star$ is updated to $i^\star \leftarrow \min(\{i^\star\} \cup \{i \,|\, \ell_i n_i \geq ck\ell_k\})$. (Note that the set of candidates $\{i \,|\, \ell_i n_i \geq ck\ell_k\}$ may change over time, as the adversary injects packets.)

Using the notion of the interesting length, we work in line with the original algorithm Greedy, with the following restrictions:

- no packet is transmitted as long as the interesting length is not determined (i.e., the set of candidates is empty);
- only a packet of length $l_{i^\star}$ can be transmitted.

As the total length of packets staying in the queue whose lengths are not interesting is at most $k \cdot ck\ell_k$, they do not have impact on the *asymptotic* value of the relative throughput. Thus, assume that there are no packets of lengths which are *not* interesting at each time t. That is, there are no packets of lengths ℓ_i such that $i \notin C$. Then, the new algorithm MGreedy works exactly as the original algorithm Greedy. The pseudo-code of algorithm MGreedy and the modified sub-routine

Algorithm 3 MGreedy

```
1: $C \leftarrow \{i \,|\, n_i\ell_i \geq ck\ell_k\}$
2: loop
3:     while $\{i \,|\, \ell_i n_i \geq ck\ell_k\} = \emptyset$ do Stay idle
4:     $C \leftarrow \{i \,|\, \ell_i n_i \geq ck\ell_k\}$
5:     $i^\star \leftarrow \min(C)$
6:     for $a = 1$ to $ck$ do $\ell' \leftarrow$ Transmit-group$(j-1)$
7:         $\ell \leftarrow \ell + \ell'$
```

Algorithm 4 Transmit-group(j)

```
1: $\ell \leftarrow 0$
2: while $\ell \leq \ell_j - \ell_{i^\star}$ do
3:     if $j > i^\star$ then $\ell' \leftarrow$ Transmit-group$(j-1)$
4:         $\ell \leftarrow \ell + \ell'$
5:     else
6:         Transmit $\ell_j$
7:         $C \leftarrow C \cup \{i \,|\, \ell_i n_i \geq ck\ell_k\}$
8:         $i^\star \leftarrow \min(C)$
9:         If a transmission of $\ell_j$ successful: $\ell \leftarrow \ell_j$
10: return $\ell$
```

Transmit-group(j), which now returns also some value ℓ, are given as Algorithm 3 and Algorithm 4, respectively.

Performance analysis of algorithm MGreedy.

We say that an execution of Transmit-group(k) is *uniform* if the algorithm transmits packets of a fixed length ℓ_i during that executions of Transmit-group(k) as well as during the executions of Transmit-group(k) directly preceding it. A new key property of algorithm MGreedy compared with Greedy is that most of its executions of sub-routine Transmit-group(k) are uniform.

Proposition 1. *At least $ck - 2k$ calls of Transmit-group in a stage of MGreedy are uniform.*

Proof. Observe that the value of $i^\star$ can only decrease during a stage and, as long as $i^\star$ remains unchanged, only packets of length $\ell_{i^\star}$ are transmitted. Therefore, the claim follows from the fact that $i^\star$ may change at most $k-1$ times during a stage. □

Now, we evaluate the relative throughput of MGreedy.

Lemma 5. *The relative throughput of the MGreedy algorithm is at least*

$$\min_{1\le j<i\le k}\left\{\frac{\lfloor\rho_{i,j}\rfloor}{\lfloor\rho_{i,j}\rfloor+\rho_{i,j}}\right\}\cdot\frac{1}{1+4/(c\eta)}\ ,$$

where η is a constant depending merely on packets' lengths.

Proof. In Lemmas 1, 2 and 3, we repeatedly use an argument that, if Greedy does not use packets of length ℓ_i for $i > m$, then each such packet transmitted by OFF corresponds to a group of (shorter) packets transmitted by Greedy of the total length ℓ_i, and this association is injective, provided Greedy is not idle at that time. For a while, assume that each execution of Transmit-group in MGreedy is uniform. Then, the above property of Greedy can be preserved for the MGreedy algorithm with a relaxation that a packet p of length ℓ_i transmitted by OFF corresponds to a group of $\lfloor\ell_i/\ell_j\rfloor$ packets of length ℓ_j transmitted by MGreedy for $j < i$. Actually, those are the packets whose successful transmission is finished during the transmission of p. The fact that there are at least $\lfloor\ell_i/\ell_j\rfloor$ such packets follows from the assumption that MGreedy is not idle at that time, its executions of Transmit-group are uniform, and the time period needed for transmission of $\lfloor\ell_i/\ell_j\rfloor$ packets ℓ_j is not larger than ℓ_i.

Let $\gamma = \min_{1\le j<i\le k}\left\{\frac{\lfloor\rho_{i,j}\rfloor}{\lfloor\rho_{i,j}\rfloor+\rho_{i,j}}\right\}$. Let $\delta_{i,j} = \frac{\rho_{i,j}}{\lfloor\rho_{i,j}\rfloor}$ and let $\delta = \max_{i>j}\{\delta_{i,j}\}$. One can check that $\gamma = 1/(1+\delta)$. This relaxation translates inequalities from Lemmas 1, 2 and 3 to:

$$\begin{array}{rcl}
\delta\cdot L_{Greedy}(\tau) & \ge & L_{OFF}(\ge m,\tau)-\ell_k\\
(1+\delta)\cdot L_{Greedy}([t_1,t]) & \ge & L_{OFF}([t_1,t])\\
 & & +q_{OFF}(<m,t)-\ell_m-\ell_k\\
(1+\delta)\cdot L_{Greedy}(\tau) & \ge & L_{OFF}(\tau)-f_m
\end{array}$$

If we apply the above inequalities instead of those from Lemmas 1, 2 and 3 in the proof of Theorem 1, we obtain the claimed result, even without the $\frac{1}{1+4/(c\eta)}$ factor, since $1/(1+\delta) = \min_{1\le j<i\le k}\left\{\frac{\lfloor\rho_{i,j}\rfloor}{\lfloor\rho_{i,j}\rfloor+\rho_{i,j}}\right\}$.

However, the above reasoning does not deal with the situation that an execution of Transmit-group is not uniform. In such a case, we cannot associate $\lfloor\ell_i/\ell_j\rfloor$ packets ℓ_j to ℓ_i transmitted by OFF in the period T such that their transmissions by MGreedy were finished in T. Fortunately, by Proposition 1, only the fraction $\frac{2k}{ck} = \frac{2}{c}$ of calls of Transmit-group(k) are not uniform. As argued earlier in the proof of Lemma 4, even without uniformity assumption we have the bound $L_{Greedy}/L_{OFF} \ge \eta$, where $\eta = \min_{i\in[2,k]}\left\{\frac{\rho_i-1}{2\rho_i-1}\right\}$.

Let us split packets transmitted by OFF by time t into those whose transmission was inside periods when MGreedy works in a uniform manner, denoted by

$L_{OFF,1}(t)$, and the remaining ones, denoted by $L_{OFF,2}(t)$. Given the fact that the fraction at least $(1-\frac{2}{c})$ of transmitted packets by MGreedy are sent in uniform way, we have the following

$$\begin{aligned}\lim_{t\to\infty}(1-2/c)\cdot L_{MGreedy}(t)/L_{OFF,1}(t) &\geq \gamma\\ \lim_{t\to\infty}(2/c)\cdot L_{MGreedy}(t)/L_{OFF,2}(t) &\geq \eta\end{aligned}$$

The second bound implies that $(2/c)\cdot L_{MGreedy}(t)/L_{OFF,2}(t)\geq \eta/2$ for each large enough t, and therefore $L_{OFF,2}(t)\leq 4/(c\cdot\eta)L_{MGreedy}(t)$. This implies also that either $L_{MGreedy}(t) > L_{OFF}(t)$ or $L_{OFF,2}(t)\leq 4/(c\cdot\eta)L_{OFF}(t)$ for sufficiently large t. As the first condition implies that the relative throughput is close to 1, assume that $L_{OFF,2}(t)\leq 4/(c\cdot\eta)L_{OFF}(t)$. As $L_{OFF}(t) = L_{OFF,1}(t)+L_{OFF,2}(t)$, this implies that

$$\lim_{t\to\infty} L_{MGreedy}(t)/L_{OFF}(t)\geq\gamma\cdot\frac{1}{1+4/(c\eta)}\ .$$

□

As we can choose arbitrarily large c, Lemma 5 implies that the relative throughput of MGreedy might be arbitrarily close to the upper bound from Theorem 2. In the following theorem, we argue that one can modify MGreedy such that it gradually increases the constant c during its execution, which guarantees the optimal relative throughput.

Theorem 3. *The optimal relative throughput of an online algorithm is equal to*

$$\min_{1\leq j<i\leq k}\left\{\frac{\lfloor\rho_{i,j}\rfloor}{\lfloor\rho_{i,j}\rfloor+\rho_{i,j}}\right\}\ .$$

Proof. Let $\gamma=\min_{1\leq j<i\leq k}\left\{\frac{\lfloor\rho_{i,j}\rfloor}{\lfloor\rho_{i,j}\rfloor+\rho_{i,j}}\right\}$. By choosing sufficiently large constant c, algorithm MGreedy can achieve the relative throughput which is arbitrarily close to γ. More precisely, it is $\gamma_c=\gamma\cdot\frac{1}{1+4/(c\eta)}$. From the proofs of Lemmas 4 and 5, one can derive a polynomial $p(1/\varepsilon,c,k,\ell_1,\ldots,\ell_k)$ such that

$$L_{MGreedy}(t)\geq(1-\varepsilon)\gamma_c L_{OFF}(t)\ ,$$

provided MGreedy transmitted packets of length at least $p(1/\varepsilon,c,k,\ell_1,\ldots,\ell_k)$. Using this bound, one can design an adaptive version of MGreedy which gradually increases the value of its parameter c. The value of c is increased to $2c$ when the total length of transmitted packets is long enough to guarantee that the current relative throughput is close enough to γ_c for the current value of c on one side, and deterioration of the relative throughput following from the increase of c in the initial part of the computation with larger c is meaningless on the other side. (Note that the increase of c may cause a temporary deterioration of the relative throughput, since larger c requires more copies of ℓ_i in the queue to consider ℓ_i as an interesting packet's length.) In this way, the relative throughput of the algorithm will get arbitrary close to γ after sufficiently long time. (The actual bound on the current relative throughput at time t depends rather on the number of successfully transmitted packets than on time t.) $\square$

One can observe that, in order to get closer to the asymptotically optimal relative throughput, our algorithms wait until there are many packets waiting for transmission in the queue (of total length at least $ck^2\ell_k$ in the worst case). That is why we conjecture that the original much simpler algorithm Greedy might turn out to be more efficient in usual real life scenarios. Therefore, from practical point of view, it is interesting to design an algorithm which achieves optimal (asymptotic) relative throughput and minimizes the maximum of the relative throughput over all times t.

4 An algorithm for a scenario with speedup

Now we return to the packets whose lengths fulfil divisibility property, i.e. $\ell_i/\ell_{i-1} \in \mathbb{N}$ for $1 < i < k$, and address the problem of increasing throughput by enabling algorithm to work with greater speed. We design an algorithm Prudent which, working with speedup $s = 2$, achieves relative throughput 1. This algorithm works in *phases*, where a phase is a time period between two consecutive errors. Behaviour of the algorithm in a phase is described as Algorithm 5. During each phase it tries to send packets of maximal length which do not exceed the total length of packets sent so far. It can be treated as a greedy strategy restricted by a "safety policy" that does not allow to send long packets unless the cost of their unsuccessful transmissions can be amortized by an advantage over an adversary gained during the earlier transmissions since the time of the last error.

Performance analysis of algorithm Prudent. Consider any offline algorithm OFF.

Lemma 6. *The total length of packets sent by Prudent is less than the total length of packets sent by OFF by no more than $5/2k\ell_k$.*

In the proof of the lemma we consider potential gain of OFF over Prudent restricted to $k - i$ longest types of packets for $i = k - 1, \dots, 0$. The proof is inductive and it is splitted into next two propositions. The first one, being the

Algorithm 5 Prudent

```
1: loop
2:     while $\{i \,|\, \ell_i n_i \geq \ell_k\} = \emptyset$ do Stay idle
3:     let $i$ be the smallest number such that: $n_i \ell_i \geq \ell_k$;
4:     if $i < k$ then
5:         transmit $\ell_{i+1}/\ell_i$ packets $\ell_i$;
6:         $L_{sent} \leftarrow \ell_{i+1}$
7:         while $L_{sent} < \ell_k$ do
8:             $j \leftarrow$ maximal number such that
9:                 $n_j \ell_j \geq \ell_k - L_{sent}$ and $\ell_j \leq L_{sent}$
10:            transmit $\ell_{j+1}/\ell_j$ packets $\ell_j$
11:            $L_{sent} \leftarrow L_{sent} + \ell_{j+1}$
12:    loop
13:        transmit longest unsent packet
```

base of the induction, shows that Prudent sends no less of the longest packets than OFF. Then, in the second proposition, we make an inductive step. Here the notion of the i-th queue means the set of packets ℓ_i waiting for transmission in the queue of algorithm Prudent.

Proposition 2. *$L_{Prudent}(k, t) \geq L_{OFF}(k, t)$ for any time t.*

Proof. Let t be the earliest time in which $L_{Prudent}(k, t) < L_{OFF}(k, t)$. There were no errors in the period $\tau = [t - \ell_k, t]$, so either Prudent has transmitted enough packets before $t - \ell_k/2$ to get willing to send packets ℓ_k or, for each $j < k$, the inequality $n_j \ell_j < \ell_k$ was satisfied at time $t - \ell_k$. In both cases Prudent would send a packet ℓ_k in τ (if then there was any in its queue), which contradicts the choice of t. □

Proposition 3. *For any time t and any $1 \leq i < k$, if*

$$L_{Prudent}(\geq j, t) \geq L_{OFF}(\geq j, t) - 5/2(k - j)\ell_k, \text{ for all } j > i$$

then

$$L_{Prudent}(\geq i, t) \geq L_{OFF}(\geq i, t) - 5/2(k - i)\ell_k.$$

Proof. Let $t_b < t$ be the beginning of the phase with time t, let $\tau = [t_b, t)$. If $|\tau| = t - t_b < \ell_i$, then OFF does not send any packet ℓ_i, and thus the thesis is trivially fulfilled. Therefore assume that $|\tau| \geq \ell_i$. We distinguish two cases:

Case 1. $L_{Prudent}(\tau) = 0$,

Case 2. $L_{Prudent}(\tau) > 0$.

Note that in the first case $|\tau| < \ell_k/2$, so $L_{OFF}(\geq i, \tau) < \ell_k/2$ and $L_{OFF}(\geq i, t) < L_{OFF}(\geq i, t_b) + \ell_k/2$. On the other hand,

$$\begin{array}{rcl} L_{Prudent}(\geq i, t) & = & L_{Prudent}(\geq i, t_b) + L_{Prudent}(\geq i, \tau) \\ & = & L_{Prudent}(> i, t_b) + L_{Prudent}(i, t_b) \; . \end{array}$$

The second equality holds since Prudent sends no packets in τ.

To estimate the right side of this equation note that Prudent has not tried to send a packet ℓ_i at t_b, so its i-th queue contained no more than ℓ_k/ℓ_i packets. Therefore $L_{Prudent}(i, t_b) \geq L_{OFF}(i, t_b) - \ell_k$. Combining this inequality with inductive bounds on $L_{Prudent}(> i, t_b)$ we get:

$$\begin{array}{rcl} L_{Prudent}(\geq i, t) & \geq & L_{OFF}(> i, t_b) - 5/2(k - (i + 1))\ell_k \\ & & +L_{OFF}(i, t_b) - \ell_k \\ & = & L_{OFF}(\geq i, t_b) + 3/2\ell_k - 5/2(k - i)\ell_k \\ & > & L_{OFF}(\geq i, t) - 5/2(k - i)\ell_k \; . \end{array}$$

Let now $L_{Prudent}(\tau) > 0$. Let packets of length l_i or longer be called *long* and let packets shorter than ℓ_i be called *short*. Since we are interested in estimating the volume of packets of length at least ℓ_i transmitted by Prudent, we check carefully what may happen after time $t_b + \ell_i/2$, i.e., at the moment when Prudent can start to transmit long packets. We analyse separately three possible situations:

Subcase 2.1. Prudent sends a small packet at some moment after $t_b + \ell_i/2$;

Subcase 2.2. after $t_b + \ell_i/2$ algorithm Prudent sends only long packets;

Subcase 2.3. after $t_b + \ell_i/2$ algorithm Prudent does not complete transmission of any packet.

First, consider *Subcase* 2.1. Let t' be the time at which Prudent for the last time before t started transmitting a packet ℓ_j for some $j < i$. Since Prudent sends packets in blocks of lengths $\ell_1, \ldots, \ell_k$ (recall that Prudent has speedup 2 and $\ell_i/\ell_{i-1} \in \mathbb{N}$), we know that $t' \geq t_b + \ell_i/2$. Therefore we can conclude that at time t' the i-th queue contained less than ℓ_k/ℓ_i packets and estimate $L_{Prudent}(i, t')$ by $L_{Prudent}(i, t') \geq L_{OFF}(i, t') - \ell_k$. This together with the inductive hypothesis gives

$$L_{Prudent}(\geq i, t') \geq L_{OFF}(\geq i, t') - 5/2(k - i)\ell_k + 3/2\ell_k \; .$$

During the period $[t', t]$, algorithm Prudent was successfully transmitting packets of length at least ℓ_i with the exception of the beginning of the period, when it was sending packet ℓ_j, and possibly the end, since its transmission of the last packet could be stopped by an error. Thus, in this period OFF could send long packets of the total length at most $\ell_j/2 + \ell_k$ larger than Prudent. So finally we have

$$\begin{aligned} L_{Prudent}(\geq i, t) &= L_{Prudent}(\geq i, t') + L_{Prudent}(\geq i, [t't]) \\ &\geq L_{OFF}(\geq i, t') - 5/2(k-i)\ell_k + 3/2\ell_k \\ &\quad + L_{OFF}(\geq i, [t'-t]) - 3/2\ell_k \\ &\geq L_{OFF}(\geq i, t) - 5/2(k-i)\ell_k \ . \end{aligned}$$

As for *Subcase* 2.2, note that in the period $\tau = [t_b, t]$, Prudent finishes successfully its last transmission after time $(t_b + t)/2$. Otherwise it would successfully send more long packets (remind that the length of each next packet chosen by Prudent does not exceed the total length of packets transmitted so far). Thus we have $L_{Prudent}(\geq i, \tau) > 2(|\tau|/2 - \ell_i/2) = |\tau| - \ell_i$. On the other hand $L_{OFF}(\geq i, \tau) \leq |\tau|$. Now we use the divisibility property of the packet lengths and observe that both $L_{Prudent}(\geq i, \tau)$ and $L_{OFF}(\geq i, \tau)$ are multiple of ℓ_i. Therefore the lower bound on the value $L_{Prudent}(\geq i, \tau)$ is not less than the upper bound on $L_{OFF}(\geq i, \tau)$, hence $L_{Prudent}(\geq i, \tau) \geq L_{OFF}(\geq i, \tau)$.

The situation described in the third subcase can not happen as it is contradictory with our assumptions $|\tau| \geq \ell_i$ and $L_{Prudent}(\tau) > 0$, which directly follow from the construction of algorithm Prudent. □

As a simple consequence of Lemma 6 we get the following theorem.

Theorem 4. *The relative throughput of Algorithm Prudent working with speed-up* 2 *is equal to* 1*, provided* $\ell_i/\ell_{i-1} \in \mathbb{N}$ *for each* $i \in [2, k]$.

5 Conclusions

We presented novel efficient and reliable algorithms for online scheduling of packets of different lengths. The first protocol assures maximum possible throughput for any arrival and jamming patterns, and additionally it guarantees to be no more than twice worse than the throughput of any other scheduling algorithm run under the same patterns. The second algorithm guarantees at least as high throughput as the optimal one, when run with additional speed-up of 2, i.e., with twice higher frequency. It demonstrates that one can use available resources in a scalable way to improve throughput for any arrival and jamming patterns, even the worst possible ones.

The considered framework is very general, and therefore it leaves a number of open extensions for further study, both theoretical, simulational and experimental. For example, what is the relative throughput in case of “average” arrival patterns, i.e., satisfying some stochastic constraints. In case of two packet lengths, it has been shown in [4] that for some stochastic distributions the relative throughput could be higher than $1/2$, and it would be interesting to give a complete characterization of stochastic arrival case for arbitrary number of packet lengths. Similarly, some restricted class of arrival and/or jamming patterns, e.g., motivated by specific physical or mobility scenarios, could allow better use of the channel. For such more specific settings, theoretical results could be also complemented by simulations run for particular physical models. Other extensions could involve packet deadlines, priorities and dependencies.

References

[1] M. Ajtai, J. Aspnes, C. Dwork, and O. Waarts. A theory of competitive analysis for distributed algorithms. In *Proceedings of the 35th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 401–411. IEEE, 1994.

[2] L. Anantharamu, B. S. Chlebus, D. R. Kowalski, and M. A. Rokicki. Online parallel scheduling of non-uniform tasks: Trading failures for energy. In *Proceedings of the 29th IEEE International Conference on Computer Communications (INFOCOM)*, pages 146–150. IEEE, 2010.

[3] M. Andrews and L. Zhang. Scheduling over a time-varying user-dependent channel with applications to high-speed wireless data. *J. ACM*, 52(5):809–834, Sept. 2005.

[4] A. F. Anta, C. Georgiou, D. R. Kowalski, J. Widmer, and E. Zavou. Measuring the impact of adversarial errors on packet scheduling strategies. In *Proceedings of the 20th International Colloquium on Structural Information and Communication Complexity (SIROCCO)*. Springer, 2013.

[5] A. F. Anta, C. Georgiou, D. R. Kowalski, and E. Zavou. Online parallel scheduling of non-uniform tasks: Trading failures for energy. In *Proceedings of the 19th International Symposium on Fundamentals of Computation Theory (FCT)*, pages 145–158. Springer, 2013.

[6] B. Awerbuch, S. Kutten, and D. Peleg. Competitive distributed job scheduling. In *Proceedings of the twenty-fourth annual ACM Symposium on Theory of Computing (STOC)*, pages 571–580. ACM, 1992.

[7] T. Jurdzinski, D. R. Kowalski, and K. Lorys. Online packet scheduling under adversarial jamming. *CoRR*, 2013.

[8] T. Kesselheim. Dynamic packet scheduling in wireless networks. In *PODC*, pages 281–290, 2012.

[9] C. Meiners and E. Torng. Mixed criteria packet scheduling. *Algorithmic Aspects in Information and Management*, pages 120–133, 2007.

[10] M. L. Pinedo. *Scheduling: theory, algorithms, and systems*. Springer, 2012.

[11] K. Pruhs, J. Sgall, and E. Torng. Online scheduling. pages 115–124. CRC Press, 2003.

[12] A. Raghavan, K. Ramchandran, and I. Kozintsev. Continuous error detection (ced) for reliable communication. *IEEE Transactions on Communications*, 49(9):1540–1549, 2001.

[13] A. Richa, C. Scheideler, S. Schmid, and J. Zhang. Competitive throughput in multi-hop wireless networks despite adaptive jamming. *Distributed Computing*, pages 1–13, 2012.

[14] D. D. Sleator and R. E. Tarjan. Amortized efficiency of list update and paging rules. *Communications of the ACM*, 28(2):202–208, 1985.