Measuring the Impact of Adversarial Errors on Packet Scheduling Strategies*

Antonio Fernández Anta[1], Chryssis Georgiou[2], Dariusz R. Kowalski[3]**, Joerg Widmer[1], and Elli Zavou[1,4]***

[1] Institute IMDEA Networks
[2] University of Cyprus
[3] University of Liverpool
[4] Universidad Carlos III de Madrid

Abstract. In this paper we explore the problem of achieving efficient packet transmission over unreliable links with worst case occurrence of errors. In such a setup, even an omniscient offline scheduling strategy cannot achieve stability of the packet queue, nor is it able to use up all the available bandwidth. Hence, an important first step is to identify an appropriate metric for measuring the efficiency of scheduling strategies in such a setting. To this end, we propose a *relative throughput* metric which corresponds to the *long term competitive ratio* of the algorithm with respect to the optimal. We then explore the impact of the error detection mechanism and feedback delay on our measure. We compare instantaneous error feedback with deferred error feedback, that requires a faulty packet to be fully received in order to detect the error. We propose algorithms for worst-case adversarial and stochastic packet arrival models, and formally analyze their performance. The relative throughput achieved by these algorithms is shown to be close to optimal by deriving lower bounds on the relative throughput of the algorithms and almost matching upper bounds for any algorithm in the considered settings. Our collection of results demonstrate the potential of using instantaneous feedback to improve the performance of communication systems in adverse environments.

1 Introduction

Motivation. Packet scheduling [8] is one of the most fundamental problems in computer networks. As packets arrive, the sender (or scheduler) needs to continuously make scheduling decisions. Typically, the objective is to maximize the *throughput* of the link or to achieve stability. Furthermore, the sender needs to take decisions without knowledge of future packet arrivals. Therefore, many times this problem is treated as an *online* scheduling problem [4, 10] and *competitive analysis* [1, 13] is used to evaluate the performance of proposed solutions: the worst-case performance of an online algorithm is compared with the performance of an offline optimal algorithm that has a priori knowledge of the problem's input.

In this work we focus on online packet scheduling over *unreliable* links, where packets transmitted over the link might be corrupted by bit errors. Such errors may, for

* This research was supported in part by the Comunidad de Madrid grant S2009TIC-1692, Spanish MICINN/MINECO grant TEC2011-29688-C02-01, and NSF of China grant 61020106002.
** This work was performed during the visit of D. Kowalski to Institute IMDEA Networks
*** Partially supported by FPU Grant from MECD

example, be caused by an increased noise level or transient interference on the link, that in the worst case could be caused by a malicious entity or an attacker. In the case of an error the affected packets must be retransmitted. To investigate the impact of such errors on the scheduling problem under study and provide *provable guarantees*, we consider the worst case occurrence of errors, that is, we consider errors caused by an omniscient and adaptive *adversary* [12]. The adversary has full knowledge of the protocol and its history, and it uses this knowledge to decide whether it will cause errors on the packets transmitted in the link at a certain time or not. Within this general framework, the packet arrival is continuous and can either be controlled by the adversary or be stochastic.

Contributions. Packet scheduling performance is often evaluated using throughput, measured in absolute terms (e.g., in bits per second) or normalized with respect to the bandwidth (maximum transmission capacity) of the link. This throughput metric makes sense for a link without errors or with random errors, where the full capacity of the link can be achieved under certain conditions. However, if adversarial bit errors can occur during the transmission of packets, the full capacity is usually not achievable by any protocol, unless restrictions are imposed on the adversary [2, 12]. Moreover, since a bit error renders a whole packet unusable (unless costly techniques like PPR [5] are used), a throughput equal to the capacity minus the bits with errors is not achievable either. As a consequence, in a link with adversarial bit errors, a fair comparison should compare the throughput of a specific algorithm to the maximum achievable amount of traffic that *any* protocol could send across the link. This introduces the challenge of identifying an appropriate metric to measure the throughput of a protocol over a link with adversarial bit errors.

Relative throughput: Our first contribution is the proposal of a *relative throughput* metric for packet scheduling algorithms under unreliable links (Section 2). This metric is a variation of the competitive ratio typically considered in online scheduling. Instead of considering the ratio of the performance of a given algorithm over that of the optimal offline algorithm, we consider the limit of this ratio as time goes to infinity. This corresponds to the *long term competitive ratio* of the algorithm with respect to the optimal.

Problem outline: We consider a sender that transmits packets to a receiver over an unreliable link, where the errors are controlled by an adversary. Regarding packet arrivals (at the sender), we consider two models: (a) the arrival times and their sizes follow a stochastic distribution, and (b) the arrival times and their sizes are also controlled by an adversary. The general offline version of our scheduling problem, in which the scheduling algorithm knows a priori when errors will occur, is NP-hard. This further motivates the need for devising simple and efficient online algorithms for the problem we consider.

Feedback mechanisms: Then, moving to the online problem requires detecting the packets received with errors, in order to retransmit them. The usual mechanism [7], which we call *deferred feedback*, detects and notifies the sender that a packet has suffered an error after the whole packet has been received by the receiver. It can be shown that, even when the packet arrivals are stochastic and packets have the same length, no online scheduling algorithm with deferred feedback can be competitive with respect to the offline one. Hence, we center our study a second mechanism, which we call *instantaneous feedback*. It detects and notifies the sender of an error the moment this

Arrivals	Feedback	Upper Bound	Lower Bound
Adversarial	Deferred	0	0
	Instantaneous	$T_{\text{Alg}} \leq \overline{\gamma}/(\gamma+\overline{\gamma})$ $T_{LL} = 0, T_{SL} \leq 1/(\gamma+1)$	$T_{SL-Pr} \geq \overline{\gamma}/(\gamma+\overline{\gamma})$
Stochastic	Deferred	0	0
	Instantaneous	$T_{\text{Alg}} \leq \overline{\gamma}/\gamma$ $T_{\text{Alg}} \leq \max\left\{\lambda p \ell_{min}, \frac{\overline{\gamma}}{(\gamma+\overline{\gamma})}\right\}$ if $p < q$ $T_{LL} = 0, T_{SL} \leq 1/(\gamma+1)$	$T_{CSL-Pr} \geq \overline{\gamma}/(\gamma+\overline{\gamma})$, if $\lambda p \ell_{min} \leq \overline{\gamma}/(2\gamma)$ $T_{CSL-Pr} \geq \min\{\lambda p \ell_{min}, \overline{\gamma}/\gamma\}$, otherwise

Table 1: Summary of results presented. The results for deferred feedback are for one packet length, while the results for instantaneous feedback are for 2 packet lengths ℓ_{min} and ℓ_{max}. Note that $\gamma = \ell_{max}/\ell_{min}$, $\overline{\gamma} = \lfloor \gamma \rfloor$, λp is the arrival rate of ℓ_{min} packets, and p and $q = 1 - p$ are the proportions of ℓ_{min} and ℓ_{max} packets, respectively.

error occurs. This mechanism can be thought of as an abstraction of the emerging Continuous Error Detection (CED) framework [11] that uses arithmetic coding to provide continuous error detection. The difference between deferred and instantaneous feedback is drastic, since for the instantaneous feedback mechanism, and for packets of the same length, it is easy to obtain optimal relative throughput of 1, even in the case of adversarial arrivals. However, the problem becomes substantially more challenging in the case of non-uniform packet lengths. Hence, we analyze the problem for the case of packets with two different lengths, ℓ_{min} and ℓ_{max}, where $\ell_{min} < \ell_{max}$.

Bounds for adversarial arrivals: We show (Section 3), that an online algorithm with instantaneous feedback can achieve at most almost half the relative throughput with respect to the offline one. It can also be shown that two basic scheduling policies, giving priority either to short *(SL – Shortest Length)* or long *(LL – Longest Length)* packets, are not efficient under adversarial errors. Therefore, we devise a new algorithm, called SL-Preamble, and show that it achieves the optimal online relative throughput. Our algorithm, transmits a "sufficiently" large number of short packets while making sure that long packets are transmitted from time to time.

Bounds for stochastic arrivals: In the case of stochastic packet arrivals (Section 4), as one might expect, we obtain better relative throughput in some cases. The results are summarized in Table 1. We propose and analyze an algorithm, called CSL-Preamble, that achieves relative throughput that is optimal. This algorithm schedules packets according to SL-Preamble, giving preference to short packets depending on the parameters of the stochastic distribution of packet arrivals. (If the distribution is not known, then one needs to use the algorithm developed for the case of adversarial arrivals that needs no knowledge a priori.) We show that the performance of algorithm CSL-Preamble is optimal for a wide range of parameters of stochastic distributions of packets arrivals, by proving a matching upper bound for the relative throughput of any algorithm in this

setting.(Analyzing algorithms yields lower bounds on the relative throughput, while analyzing adversarial strategies yields upper bounds on the relative throughput.)

A note on randomization: All the proposed algorithms are deterministic. Interestingly, it can be shown that using randomization does not improve the results; the upper bounds already discussed hold also for the randomized case.

To the best of our knowledge, this is the first work that investigates in depth the impact of adversarial worst-case link errors on the throughput of the packet scheduling problem. Collectively, our results (see Table 1) show that instantaneous feedback can achieve a significant relative throughput under worst-case adversarial errors (almost half the relative throughput that the offline optimal algorithm can achieve). Furthermore, we observe that in some cases, stochastic arrivals allow for better performance.

Omitted results, proofs and discussion can be found in the full version [3].

Related work. A vast amount of work exists for online (packet) scheduling. Here we focus only on the work that is most related to ours. For more information the reader can consult [9] and [10]. The work in [6] considers the packet scheduling problem in wireless networks. Like our work, it looks at both stochastic and adversarial arrivals. Unlike our work though, it considers only *reliable* links. Its main objective is to achieve maximal throughput guaranteeing *stabiliy*, meaning bounded time from injection to delivery. The work in [2] considers online packet scheduling over a wireless channel, where both the channel conditions and the data arrivals are governed by an adversary. Its main objective is to design scheduling algorithms for the base-station to achieve stability in terms of the size of queues of each mobile user. Our work does not focus on stability, as we assume errors controlled by an unbounded adversary that can always prevent it. The work in [12] considers the problem of devising local access control protocols for wireless networks with a single channel, that are provably robust against *adaptive adversarial jamming*. At certain time steps, the adversary can jam the communication in the channel so that the wireless nodes do not receive messages (unlike our work, where the receiver might receive a message, but it might contain bit errors). Although the model and the objectives of this line of work is different from ours, it shares the same concept of studying the impact of adversarial behavior on network communication.

2 Model

Network setting. We consider a sending station transmitting packets over a link. Packets arrive at the sending station continuously and may have different lengths. Each packet that arrives is associated with a length and its arrival time (based on the station's local clock). We denote by ℓ_{min} and ℓ_{max} the smallest and largest lengths, respectively, that a packet may have. We use the notation $\gamma = \ell_{max}/\ell_{min}$, $\overline{\gamma} = \lfloor \gamma \rfloor$ and $\hat{\gamma} = \lceil \gamma \rceil - 1$. The link is unreliable, that is, transmitted packets might be corrupted by bit errors. We assume that all packets are transmitted at the same bit rate, hence the transmission time is proportional to the packet's length.

Arrival models. We consider two models for packet arrivals.

Adversarial: 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.

Stochastic: We consider a probabilistic distribution D_a, under which packets arrive at the sending station and a probabilistic distribution D_s, for the length of the packets. In particular, we assume packets arriving according to a Poisson process with parameter $\lambda > 0$. When considering two packet lengths, ℓ_{min} and ℓ_{max}, each packet that arrives is assigned one of the two lengths independently, with probabilities $p > 0$ and $q > 0$ respectively, where $p + q = 1$.

Packet bit errors. We consider an adversary that controls the bit errors of the packets transmitted over the link. An adversarial error pattern is defined as a collection of error events on the link caused by the adversary. More precisely, an error event at time t specifies that an instantaneous error occurs on the link at time t, so the packet that happens to be on the link at that time is corrupted with bit errors. A corrupted packet transmission is unsuccessful, therefore the packet needs to be retransmitted in full. As mentioned before, we consider an *instantaneous feedback* mechanism for the notification of the sender about the error. The instant the packet suffers a bit error the sending station is notified (hence it can stop transmitting the remainder of the packet, if any).

The power of the adversary. Adversarial models are typically used to argue about the algorithm's behavior in worst-case scenarios. In this work we assume an adaptive adversary that knows the algorithm and the history of the execution up to the current point in time. In the case of stochastic arrivals, this includes all stochastic packet arrivals up to this point, and the length of the packets that have arrived. However it only knows the distribution but neither the exact timing nor the length of the packets arriving beyond the current time.

Note that in the case of deterministic algorithms, in the model of adversarial arrivals the adversary has full knowledge of the computation, as it controls both packet arrivals and errors, and can simulate the behavior of the algorithm in the future (there are no random bits involved in the computation). This is not the case in the model with stochastic arrivals, where the adversary does not control the timing of future packet arrivals, but knows only about the packet arrival and length distributions.

Efficiency metric: *Relative throughput.* Due to dynamic packet arrivals and adversarial errors, the real link capacity may vary throughout the execution. Therefore, we view the problem of packet scheduling in this setting as an online problem and we pursue long-term competitive analysis. Specifically, let A be an arrival pattern and E an error pattern. For a given deterministic algorithm Alg, let $L_{\text{Alg}}(A, E, t)$ be the total length of all the successfully transferred (i.e., non-corrupted) packets by time t under patterns A and E. Let OPT be the offline optimal algorithm that knows the exact arrival and error patterns before the start of the execution. We assume that OPT devises an optimal schedule that maximizes at each time t the successfully transferred packets $L_{\text{OPT}}(A, E, t)$. Observe that, in the case of stochastic arrivals, the worst-case adversarial error pattern may depend on stochastic injections. Therefore, we view E as a function of an arrival pattern A and time t. In particular, for an arrival pattern A we consider a function $E(A, t)$ that defines errors at time t based on the behavior of a given algorithm Alg under the arrival pattern A up to time t and the values of function $E(A, t')$ for $t' < t$.

Let $\mathcal{A}$ denote a considered arrival model, i.e., a set of arrival patterns in case of adversarial, or a distribution of packet injection patterns in case of stochastic, and let

$\mathcal{E}$ denote the corresponding adversarial error model, i.e., a set of error patterns derived by the adversary, or a set of functions defining the error event times in response to the arrivals that already took place in case of stochastic arrivals. In case of adversarial arrivals, we require that any pair of patterns $A \in \mathcal{A}$ and $E \in \mathcal{E}$ occurring in an execution must allow non-trivial communication, i.e., the value of $L_{\text{OPT}}(A, E, t)$ in the execution is unbounded with t going to infinity. In case of stochastic arrivals, we require that any adversarial error function $E \in \mathcal{E}$ applied in an execution must allow non-trivial communication for any stochastic arrival pattern $A \in \mathcal{A}$.

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

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

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

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

$$T_{\text{Alg}} = \inf_{A \in \mathcal{A}, E \in \mathcal{E}} \lim_{t \to \infty} T_{\text{Alg}}(A, E, t) \ ,$$

while in the stochastic arrival model it needs to take into account the random distribution of arrival patterns in $\mathcal{A}$, and is defined as follows:

$$T_{\text{Alg}} = \inf_{E \in \mathcal{E}} \lim_{t \to \infty} \mathbb{E}_{A \in \mathcal{A}}[T_{\text{Alg}}(A, E, t)] \ .$$

To prove lower bounds on relative throughput, we compare the performance of a given algorithm with that of OPT. When deriving upper bounds, it is not necessary to compare the performance of a given algorithm with that of OPT, but instead, with the performance of some carefully chosen offline algorithm OFF. As we demonstrate later, this approach leads to accurate upper bound results.

Finally, we consider *work conserving* online scheduling algorithms, in the following sense: as long as there are pending packets, the sender does not cease to schedule packets. Note that it does not make any difference whether one assumes that offline algorithms are work-conserving or not, since their throughput is the same in both cases (a work conserving offline algorithm always transmits, but stops the ongoing transmission as soon as an error occurs and then continues with the next packet). Hence for simplicity we do not assume offline algorithms to be work conserving.

3 Adversarial Arrivals

This section focuses on adversarial packet arrivals. First, observe that it is relatively easy and efficient to handle packets of only one length.

Proposition 1. *Any work conserving online scheduling algorithm with instantaneous feedback has optimal relative throughput of 1 when all packets have the same length.*

3.1 Upper Bound for at least Two Packet Lengths

Let Alg be any deterministic algorithm for the considered packet scheduling problem. In order to prove upper bounds, Alg will be competing with an offline algorithm OFF. The scenario is as follows. We consider an infinite supply of packets of length ℓ_{max} and initially assume that there are no packets of length ℓ_{min}. We define as a *link error event*, the point in time when the adversary corrupts (causes an error to) any packet

that happens to be in the link at that specific time. We divide the execution in *phases*, defined as the periods between two consecutive link error events. We distinguish 2 types of phases as described below and give a description for the behavior of the adversarial models $\mathcal{A}$ and $\mathcal{E}$. The adversary controls the arrivals of packets at the sending station and error events of the link, as well as the actions of algorithm OFF. The two types of phases are as follows:

1. a phase in which Alg starts by transmitting an ℓ_{max} packet (the first phase of the execution belongs to this class). Immediately after Alg starts transmitting the ℓ_{max} packet, a set of $\hat{\gamma}$ ℓ_{min} packets arrive, that are scheduled and transmitted by OFF. After OFF completes the transmission of these packets, a link error occurs, so Alg cannot complete the transmission of the ℓ_{max} packet (more precisely, the packet undergoes a bit error, so it needs to be retransmitted). Here we use the fact that $\hat{\gamma} < \gamma$.
2. a phase in which Alg starts by transmitting an ℓ_{min} packet. In this case, OFF transmits an ℓ_{max} packet. Immediately after this transmission is completed, a link error occurs. Observe that in this phase Alg has transmitted successfully several ℓ_{min} packets (up to $\overline{\gamma}$ of them).

Let A and E be the specific adversarial arrival and error patterns in an execution of Alg. Let us consider any time t (at the end of a phase for simplicity) in the execution. Let v_1 be the number of phases of type 1 executed by time t. Similarly, let $v_2(j)$ be the number of phases of type 2 executed by time t in which Alg transmits j ℓ_{min} packets, for $j \in [1, \overline{\gamma}]$. Then, the relative throughput can be computed as follows.

$$T_{\text{Alg}}(A, E, t) = \frac{\ell_{min} \sum_{j=1}^{\overline{\gamma}} j v_2(j)}{\ell_{max} \sum_{j=1}^{\overline{\gamma}} v_2(j) + \ell_{min} \hat{\gamma} v_1}. \quad (1)$$

From the arrival pattern A, the number of ℓ_{min} packets injected by time t is exactly $\hat{\gamma} v_1$. Hence, $\sum_{j=1}^{\overline{\gamma}} j v_2(j) \leq \hat{\gamma} v_1$. It can be easily observed from Eq. 1 that the relative throughput increases with the average number of ℓ_{min} packets transmitted in the phases of type 2. Hence, the throughput would be maximal if all the ℓ_{min} packets are used in phases of type 2 with $\overline{\gamma}$ packets. With the above we obtain the following theorem.

Theorem 1. *The relative throughput of Alg under adversarial patterns A and E and up to time t is at most $\frac{\overline{\gamma}}{\gamma+\overline{\gamma}} \leq \frac{1}{2}$ (the equality holds iff γ is an integer).*

3.2 Lower Bound and SL-Preamble Algorithm

Two natural scheduling policies one could consider are the *Shortest Length* (SL) and *Longest Length* (LL) algorithms; the first gives priority to ℓ_{min} packets, whereas the second gives priority to the ℓ_{max} packets. However, these two policies are not efficient in the considered setting; LL cannot achieve a relative throughput more than 0 while SL achieves at most $T = \frac{1}{\gamma+1}$. Therefore, we present algorithm SL-Preamble that tries to combine, in a graceful and efficient manner, these two policies.

Algorithm description: At the beginning of the execution and whenever the sender is (immediately) notified by the instantaneous feedback mechanism that a link error occurred, it checks the queue of pending packets to see whether there are at least $\overline{\gamma}$ packets of length ℓ_{min} available for transmission. If there are, then it schedules $\overline{\gamma}$ of them — this is called a *preamble* — and then the algorithm continues to schedule packets using the LL policy. Otherwise, if there are not enough ℓ_{min} packets available, it simply schedules packets following the LL policy.

Algorithm analysis (sketch): We show that algorithm SL-Preamble achieves a relative throughput that matches the upper bound shown in the previous subsection, and hence, it is optimal. According to the algorithm there are four types of phases that may occur.

1. Phase starting with ℓ_{min} packet and has length $L < \overline{\gamma}\ell_{min}$
2. Phase starting with ℓ_{min} packet and length $L \geq \overline{\gamma}\ell_{min}$
3. Phase starting with ℓ_{max} packet and has length $L < \ell_{max}$
4. Phase starting with ℓ_{max} packet and length $L \geq \ell_{max}$

For phases of type 1, SL-Preamble is not able to transmit successfully the $\overline{\gamma}$ packets ℓ_{min} of the preamble, but clearly OPT is only able to complete at most as much *work* (understood as the total length of sent packets). For phases of type 2 and 4, the amount of work completed by OPT can be at most the work completed by SL-Preamble plus ℓ_{max} (and hence the former is at most twice the latter). In the case of phases of type 3, SL-Preamble is not able to successfully transmit any packet, whereas OPT might transmit up to $\hat{\gamma}\ell_{min}$ packets. Amortizing the work completed by OPT in these phases with those completed in the preambles of types 1 and 2 by algorithm SL-Preamble is the most challenging part of the proof. This process is divided into two cases, depending on whether the number of type 3 phases is bounded or not, leading to the following:

Theorem 2. *The relative throughput of Algorithm SL-Preamble is at least* $\frac{\overline{\gamma}}{\gamma+\overline{\gamma}}$.

4 Stochastic Arrivals

We now turn our attention to stochastic packet arrivals.

4.1 Upper Bounds for at least Two Packet Lengths

In order to find the upper bound of the relative throughput, we consider again an arbitrary work conserving algorithm Alg. Recall that we assume that $\lambda p > 0$ and $\lambda q > 0$, which implies that there are in fact injections of packets of both lengths ℓ_{min} and ℓ_{max} (recall the definitions of λ, p and q from Section 2). We define the following adversarial error model $\mathcal{E}$.

1. When Alg starts a phase by transmitting an ℓ_{max} packet then,
   (a) If OFF has ℓ_{min} packets pending, then the adversary extends the phase so that OFF can transmit successfully as many ℓ_{min} packets as possible, up to $\hat{\gamma}$. Then, it ends the phase so that Alg does not complete the transmission of the ℓ_{max} packet (since $\hat{\gamma}\ell_{min} < \ell_{max}$).
   (b) If OFF does not have any ℓ_{min} packets pending, then the adversary inserts a link error immediately (say after infinitesimally small time ϵ).
2. When Alg starts a phase by transmitting an ℓ_{min} packet then,
   (a) If OFF has a packet of length ℓ_{max} pending, then the adversary extends the phase so OFF can transmit an ℓ_{max} packet. By the time this packet is successfully transmitted, the adversary inserts an error and finishes the phase. Observe that in this case Alg was able to successfully transmit up to $\overline{\gamma}$ packets ℓ_{min}.
   (b) If OFF has no ℓ_{max} packets pending, then the adversary inserts an error immediately and ends the phase.

Observe that in phases of type 1b and 2b, neither OFF nor Alg are able to transmit any packet. These phases are just used by the adversary to wait for the conditions required by phases of type 1a and 2a to hold. In those latter types some packets are successfully transmitted (at least by OFF). Hence we call them *productive* phases. Analyzing a possible execution, in addition to the concept of phase that we have already used, we define *rounds*. There is a round associated with each productive phase. The round ends when its corresponding productive phase ends, and starts at the end of the prior round (or at the start of the execution if no prior round exists). Depending on the type of productive phase they contain, rounds can be classified as type 1a or 2a.

Let us fix some (large) time t. We denote by $r_{1a}^{(j)}$ the number of rounds of type 1a in which $j \leq \hat{\gamma}\ \ell_{min}$ packets are sent by OFF completed by time t. The value $r_{2a}^{(j)}$ with $j \leq \overline{\gamma}\ \ell_{min}$ packets sent by Alg, is defined similarly for rounds of type 2a. (Here rounding effects do not have any significant impact, since they will be compensated by the assumption that t is large.) We assume that t is a time when a round finishes. Let us denote by r the total number or rounds completed by time t, i.e., $\sum_{j=1}^{\overline{\gamma}} r_{2a}^{(j)} + \sum_{j=1}^{\hat{\gamma}} r_{1a}^{(j)} = r$. The relative throughput by time t can be computed as

$$T_{\text{Alg}}(A, E, t) = \frac{\ell_{min} \sum_{j=1}^{\overline{\gamma}} j \cdot r_{2a}^{(j)}}{\ell_{max} \sum_{j=1}^{\overline{\gamma}} r_{2a}^{(j)} + \ell_{min} \sum_{j=1}^{\hat{\gamma}} j \cdot r_{1a}^{(j)}}\ . \tag{2}$$

From this expression, we can show the following result.

Theorem 3. *No algorithm Alg has relative throughput larger than* $\frac{\overline{\gamma}}{\gamma}$.

Proof. It can be observed in Eq. 2 that, for a fixed r, the lower the value of $r_{1a}^{(j)}$ the higher the relative throughput. Regarding the values $r_{2a}^{(j)}$, the throughput increases when there are more rounds in the larger values of j. E.g., under the same conditions, a configuration with $r_{2a}^{(j)} = k_1$ and $r_{2a}^{(j+1)} = k_2$, has lower throughput than one with $r_{2a}^{(j)} = k_1 - 1$ and $r_{2a}^{(j+1)} = k_2 + 1$. Then, the throughput is maximized when $r_{2a}^{(\overline{\gamma})} = r$ and the rest of values $r_{1a}^{(j)}$ and $r_{2a}^{(j)}$ are 0, which yields the bound. ∎

To provide tighter bounds for some special cases, we prove the following lemma.

Lemma 1. *Consider any two constants* η, η' *such that* $0 < \eta < \lambda < \eta'$. *Then:*

(a) there is a constant $c > 0$, *dependent only on* λ, p, η, *such that for any time* $t \geq \ell_{min}$, *the number of packets of length* ℓ_{min} *(resp.,* ℓ_{max}*) injected by time* t *is* at least $t\eta p$ *(resp.,* $t\eta q$*) with probability at least* $1 - e^{-ct}$;

(b) there is a constant $c' > 0$, *dependent only on* λ, p, η', *such that for any time* $t \geq \ell_{min}$, *the number of packets of length* ℓ_{min} *(resp.,* ℓ_{max}*) injected by time* t *is* at most $t\eta' p$ *(resp.,* $t\eta' q$*) with probability at least* $1 - e^{-c't}$.

Now we can show the following result.

Theorem 4. *Let* $p < q$. *Then, the relative throughput of any algorithm Alg is at most* $\min\left\{\max\left\{\lambda p \ell_{min}, \frac{\overline{\gamma}}{\gamma+\overline{\gamma}}\right\}, \frac{\overline{\gamma}}{\gamma}\right\}$.

Proof. The claim has two cases. In the first case, $\lambda p \ell_{min} \geq \frac{\overline{\gamma}}{\gamma}$. In this case, the upper bound of $\frac{\overline{\gamma}}{\gamma}$ is provided by Theorem 3. In the second case $\lambda p \ell_{min} < \frac{\overline{\gamma}}{\gamma}$. For this case, define two constants η, η' such that $0 < \eta < \lambda < \eta'$ and $\eta' p < \eta q$. Observe that these constants always exist. Then, we prove that the relative throughput of any algorithm Alg in this case is at most $\max\left\{\eta' p \ell_{min}, \frac{\overline{\gamma}}{\gamma+\overline{\gamma}}\right\}$.

Let us introduce some notation. We use $a_t^{\min}$ and $a_t^{\max}$ to denote the number of ℓ_{min} and ℓ_{max} packets, respectively, injected up to time t. Let r_t^{off} and s_t^{off} be the number of ℓ_{max} and ℓ_{min} packets respectively, successfully transmitted by OFF by time t. Similarly, let s_t^{alg} be the number of ℓ_{min} packets transmitted by algorithm Alg by time t. Observe that $s_t^{alg} \geq r_t^{off} \geq \lfloor \frac{s_t^{alg}}{\overline{\gamma}} \rfloor$.

Let us consider a given execution and the time instants at which the queue of OFF is empty of ℓ_{min} packets in the execution. We consider two cases.

Case 1: For each time t, there is a time $t' > t$ at which OFF has the queue empty of ℓ_{min} packets. Let us fix a value $\delta > 0$ and define time instants $t_0, t_1, \ldots$ as follows. t_0 is the first time instant no smaller than ℓ_{min} at which OFF has no ℓ_{min} packet and such that $a_{t_0}^{\min} > \ell_{max}$. Then, for $i > 0$, t_i is the first time instant not smaller than $t_{i-1} + \delta$ at which OFF has no ℓ_{min} packets. The relative throughput at time t_i can be bounded as

$$T_{\text{Alg}}(A, E, t_i) \leq \frac{s_{t_i}^{alg} \ell_{min}}{r_{t_i}^{off} \ell_{max} + a_{t_i}^{\min} \ell_{min}} \leq \frac{s_{t_i}^{alg} \ell_{min}}{\lfloor \frac{s_{t_i}^{alg}}{\overline{\gamma}} \rfloor \ell_{max} + a_{t_i}^{\min} \ell_{min}}.$$

This bound grows with $s_{t_i}^{alg}$ when $a_{t_i}^{\min} > \ell_{max}$, which leads to a bound on the relative throughput as follows:

$$T_{\text{Alg}}(A, E, t_i) \leq \frac{a_{t_i}^{\min} \ell_{min}}{a_{t_i}^{\min}(\frac{\ell_{max}}{\overline{\gamma}} + \ell_{min}) - \ell_{max}} = \frac{a_{t_i}^{\min} \overline{\gamma}}{a_{t_i}^{\min}(\gamma + \overline{\gamma}) - \gamma\overline{\gamma}}.$$

Which as i goes to infinity yields a bound of $\frac{\overline{\gamma}}{\gamma+\overline{\gamma}}$.

Case 2: There is a time t_* after which OFF never has the queue empty of ℓ_{min} packets. Recall that for any $t \geq \ell_{min}$, from Lemma 1, we have that the number of ℓ_{min} packets injected by time t satisfy $a_t^{\min} > \eta' p t$ with probability at most $\exp(-c't)$ and the injected $\max$ packets satisfy $a_t^{\max} < \eta q t$ with probability at most $\exp(-ct)$. By the assumption of the theorem and the definition of η and η', $\eta' p < \eta q$. Let us define $t^* = 1/(\eta q - \eta' p)$. Then, for all $t \geq t^*$ it holds that $a_t^{\max} \geq a_t^{\min} + 1$, with probability at least $1 - \exp(-c't) - \exp(-ct)$. If this holds, it implies that OFF will always have ℓ_{max} packets in the queue.

Let us fix a value $\delta > 0$ and define $t_0 = \max(t_*, t^*)$, and the sequence of instants $t_i = t_0 + i\delta$, for $i = 0, 1, 2, \ldots$. By the definition of t_0, at all times $t > t_0$ OFF is successfully transmitting packets. Using Lemma 1, we can also claim that in the interval $(t_0, t_i]$ the probability that more than $\eta' p i \delta$ packets ℓ_{min} are injected is no more than $\exp(-c'' i\delta)$.

With the above, the relative throughput at any time t_i for $i \geq 0$ can be bounded as

$$T_{\text{Alg}}(A, E, t_i) \leq \frac{(a_{t_0}^{\min} + \eta' p \cdot i\delta)\ell_{min}}{r_{t_0}^{off} \ell_{max} + s_{t_0}^{off} \ell_{min} + i\delta}$$

with probability at least $1 - \exp(-ct_i) - \exp(-c't_i) - \exp(-c''t_i)$. Observe that as i goes to infinity the above bound converges to $\eta' p \ell_{min}$, while the probability converges exponentially fast to 1. ∎

4.2 Lower Bound and Algorithm CSL-Preamble

In this section we consider algorithm CSL-Preamble (stands for Conditional SL-Preamble), which builds on algorithm SL-Preamble presented in Section 3.2, in order to solve packet scheduling in the setting of stochastic packet arrivals. The algorithm, depending on the arrival distribution, either follows the SL policy (giving priority to ℓ_{min} packets) or algorithm SL-Preamble. More precisely, algorithm CSL-Preamble acts as follows:

If $\lambda p \ell_{min} > \frac{\overline{\gamma}}{2\gamma}$ then algorithm SL is run,
otherwise algorithm SL-Preamble is executed.

Theorem 5. *The relative throughput of algorithm CSL-Preamble is not smaller than* $\frac{\overline{\gamma}}{\gamma+\overline{\gamma}}$ *for* $\lambda p \ell_{min} \leq \frac{\overline{\gamma}}{2\gamma}$*, and not smaller than* $\min\left\{\lambda p \ell_{min}, \frac{\overline{\gamma}}{\gamma}\right\}$ *otherwise.*

Proof. (Sketch) We break the analysis of the algorithm into cases according to the probability of ℓ_{min} packet arrivals and consider the time line of executions ignoring any OPT-unproductive periods.

Case $\lambda p \ell_{min} \leq \frac{\overline{\gamma}}{2\gamma}$**.** In this case algorithm CSL-Preamble runs algorithm SL-Preamble, achieving, per Theorem 2, relative throughput of at least $\frac{\overline{\gamma}}{\gamma+\overline{\gamma}}$ under *any* error pattern.

Case $\frac{\overline{\gamma}}{2\gamma} \leq \lambda p \ell_{min} \leq 1$**.** It can be proved that the relative throughput is not smaller than $\min\left\{\eta p \ell_{min}, \frac{\overline{\gamma}}{\gamma}\right\}$, for any η satisfying $\lambda/2 < \eta < \lambda$. To prove it, we consider time points t_i being multiples of ℓ_{max} and show that with high probability, at those points there have already arrived at least $t_i \eta p$ packets. Using this property, we show that the relative throughput at time t_j is at least $\min\left\{\eta p \ell_{min} - \frac{\overline{\gamma}\ell_{min}}{t_j}, (1 - 1/\sqrt{j}) \cdot \frac{\overline{\gamma}}{\gamma}\right\}$ with probability at least $1 - c' \exp\left(-ct_{\sqrt{j}}\right)$, for some constant $c, c' > 0$ dependent only on λ, η, p. It follows that if j grows to infinity, we obtain the desired relative throughput.

Case $\lambda p \ell_{min} > 1$**.** In this case we simply observe that we get at least the same relative throughput as in case $\lambda p \ell_{min} = 1$, because we are dealing with executions saturated with packets of length ℓ_{min} with probability converging to 1 exponentially fast. (Recall that we use the same algorithm SL in the specification of CSL-Preamble, both for $\lambda p \ell_{min} = 1$ and for $\lambda p \ell_{min} > 1$.) Consequently, the relative throughput in this case is at least $\min\{\eta p \ell_{min}, \overline{\gamma}/\gamma\}$, for any $\lambda/2 < \eta < \lambda$, and thus it is at least $\min\{\lambda p \ell_{min}, \overline{\gamma}/\gamma\} \geq \min\{1, \overline{\gamma}/\gamma\} = \overline{\gamma}/\gamma$.

Combining the three cases, we get the claimed result. ∎

Observe that if we compare the upper bounds on relative throughput shown in the previous subsection with the lower bounds of the above theorem, then we may conclude that in the case where γ is an integer, algorithm CSL-Preamble is optimal (wrt relative throughput). In the case where γ is not an integer, there is a small gap between the upper and lower bound results.

5 Conclusions

This work has considered packet scheduling with dynamic packet arrivals and adversarial bit errors. We studied scenarios with two different packet lengths, developed efficient algorithms, and proved upper and lower bounds for relative throughput in average-case (i.e., stochastic) and worst-case (i.e., adversarial) online packet arrivals. These results

demonstrate that exploring instantaneous feedback mechanisms (and developing more effective implementations of it) has the potential to significantly increase the performance of communication systems.

Several future research directions emanate from this work. Some of them concern the exploration of variants of the model considered, for example, assuming that packets that suffer errors are not retransmitted (which applies when Forward Error Correction [11] is used), considering packets of more than two lengths, or assuming bounded buffers. Other lines of work deal with adding QoS requirements to the problem, such as requiring fairness in the transmission of the packets from different flows or imposing deadlines to the packets. In the considered adversarial setting, it is easy to see that even an omniscient offline solution cannot achieve stability: for example, the adversary could prevent any packet from being transmitted correctly. Therefore, an interesting extension of our work is to study conditions (e.g., restrictions on the adversary) under which an online algorithm could maintain stability, and still be efficient with respect to relative throughput. Finally, we believe that the definition of relative throughput as proposed here can be adapted, possibly in a different context, to other metrics and problems.

References

1. Miklos Ajtai, James Aspnes, Cynthia Dwork, and Orli Waarts. A theory of competitive analysis for distributed algorithms. In *Foundations of Computer Science, 1994 Proceedings., 35th Annual Symposium on*, pages 401–411. IEEE, 1994.
2. Matthew Andrews and Lisa Zhang. Scheduling over a time-varying user-dependent channel with applications to high-speed wireless data. *J. ACM*, 52(5):809–834, September 2005.
3. Antonio Fernandez Anta, Chryssis Georgiou, Dariusz R. Kowalski, Joerg Widmer, and Elli Zavou. Measuring the impact of adversarial errors on packet scheduling strategies. In ArXiv, 2013.
4. Baruch Awerbuch, Shay Kutten, and David Peleg. Competitive distributed job scheduling. In *Proceedings of the twenty-fourth annual ACM symposium on Theory of computing*, pages 571–580. ACM, 1992.
5. Kyle Jamieson and Hari Balakrishnan. Ppr: partial packet recovery for wireless networks. In *Proceedings of the 2007 conference on Applications, technologies, architectures, and protocols for computer communications*, SIGCOMM '07, pages 409–420, New York, NY, USA, 2007. ACM.
6. Thomas Kesselheim. Dynamic packet scheduling in wireless networks. In *PODC*, pages 281–290, 2012.
7. Shu Lin and Daniel J Costello. *Error control coding*, volume 123. Prentice-hall Englewood Cliffs, NJ, 2004.
8. Chad Meiners and Eric Torng. Mixed criteria packet scheduling. *Algorithmic Aspects in Information and Management*, pages 120–133, 2007.
9. Michael L Pinedo. *Scheduling: theory, algorithms, and systems*. Springer, 2012.
10. Kirk Pruhs, Eric Torng, et al. Online scheduling. 2007.
11. Anand Raghavan, Kannan Ramchandran, and Igor Kozintsev. Continuous error detection (ced) for reliable communication. *IEEE Transactions on Communications*, 49(9):1540–1549, 2001.
12. Andrea Richa, Christian Scheideler, Stefan Schmid, and Jin Zhang. Competitive throughput in multi-hop wireless networks despite adaptive jamming. *Distributed Computing*, pages 1–13, 2012.
13. Daniel D Sleator and Robert E Tarjan. Amortized efficiency of list update and paging rules. *Communications of the ACM*, 28(2):202–208, 1985.