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 [7] 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 [3, 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 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

*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.

†Partially supported by FPU Grant from MECD

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	$T_{\text{Alg}} \leq \overline{\gamma}/\gamma$ $T_{\text{Alg}} \leq \max\{\lambda p \ell_{min}, \overline{\gamma}/(\gamma+\overline{\gamma})\}$, 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.

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 [4] 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 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 [1]. This further motivates the need for devising simple and efficient online algorithms for the problem we consider.

[1] Some of the results are omitted due to space limitation and can be found in the Appendix.

Feedback mechanisms: Then, moving to the online problem requires detecting the packets received with errors, in order to retransmit them. The usual mechanism [6], 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 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[1]. 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 the matching upper bound[2] for the relative throughput of any algorithm in this setting.

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. For more details see Appendix D.

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.

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 [5] 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

[1] If the distribution is not known, then obviously one needs to use the algorithm developed for the case of adversarial arrivals that needs no knowledge a priori.

[2] Analyzing algorithms yields lower bounds on the relative throughput, while analyzing adversarial strategies yields upper bounds on the relative throughput.

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 in such a way 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 (and 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 algorithm 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.*

Proof: Consider an algorithm Alg. Since it is work conserving, as long as there are pending packets, it schedules them. If an error is reported by the feedback mechanism, the algorithm simply retransmits another (or the same) packet. Since the notification is instantaneous, it is not difficult to see that the a priori knowledge that the offline optimal algorithm has, does not help in transmitting more non-corrupted packets than Alg. ■

3.1 Upper Bound

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 p_1 be the number of phases of type 1 executed by time t. Similarly, let $p_2(j)$ 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.

$$T_{\text{Alg}}(A, E, t) = \frac{\ell_{min} \sum_{j=1}^{\overline{\gamma}} j p_2(j)}{\ell_{max} \sum_{j=1}^{\overline{\gamma}} p_2(j) + \ell_{min} \hat{\gamma} p_1}. \tag{1}$$

From the arrival pattern A, the number of ℓ_{min} packets injected by time t is exactly $\hat{\gamma} p_1$. Hence, $\sum_{j=1}^{\overline{\gamma}} j p_2(j) \leq \hat{\gamma} p_1$. 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).*

Proof: Applying the bound $\sum_{j=1}^{\overline{\gamma}} p_2(j) \geq \sum_{j=1}^{\overline{\gamma}} \frac{jp_2(j)}{\overline{\gamma}}$ in Eq (1), we get

$$T_{\text{Alg}}(A,E,t) \leq \frac{\ell_{min} \sum_{j=1}^{\overline{\gamma}} jp_2(j)}{\frac{\ell_{max}}{\overline{\gamma}} \sum_{j=1}^{\overline{\gamma}} jp_2(j) + \ell_{min}\hat{\gamma}p_1},$$

which is a function that increases with $\sum_{j=1}^{\overline{\gamma}} jp_2(j)$. Since $\sum_{j=1}^{\overline{\gamma}} jp_2(j) \leq \hat{\gamma}p_1$, the relative throughput can be bounded as

$$T_{\text{Alg}}(A,E,t) \quad \leq \quad \frac{\ell_{min}\overline{\gamma}\hat{\gamma}p_1/\overline{\gamma}}{\ell_{max}\frac{\hat{\gamma}p_1}{\overline{\gamma}} + \ell_{min}\hat{\gamma}p_1} = \frac{\ell_{min}\overline{\gamma}}{\ell_{max} + \ell_{min}\overline{\gamma}} = \frac{\overline{\gamma}}{\gamma+\overline{\gamma}}.$$

■

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: 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. Let us define two types of time periods for the link in the executions of algorithm SL-Preamble: the *active* and the *inactive* periods. An active period is one in which the link experiences no errors and SL-Preamble has pending packets waiting to be transferred, whereas an inactive one is such that either the link has an error point or the queue of pending packets is empty for SL-Preamble. In the case of inactive periods, note that, if the link has an error, neither SL-Preamble nor OPT can make any progress in transmitting an error-free packet. Similarly, if the queue of pending packets is empty for SL-Preamble, it must be empty for OPT as well (otherwise it would contradict the optimality of OPT). Hence, we look at the active periods, which we refer to as *phases*, and according to the above algorithm we observe that 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 $L \geq \ell_{max}$

We now introduce some notation that will be used throughout the analysis. For the execution of SL-Preamble and within the ith phase, let a_i be the number of successfully transmitted ℓ_{min} packets not in the preambles, b_i the number of successfully transmitted ℓ_{max} packets, and c_i the number of successfully transmitted ℓ_{min} packets in preambles. For the execution of OPT and within the ith phase, let a_i^* be the total number of successfully transmitted ℓ_{min} packets and b_i^* the total number of successfully transmitted ℓ_{max} packets. Let $C_A^j(i)$ and $C_O^j(i)$ denote the total amount successfully transmitted within a phase i of type j by SL-Preamble and OPT, respectively.

Analyzing the different types of phases we make some observations. First, for phases of type 1, SL-Preamble is not able to transmit successfully the $\overline{\gamma}\,\ell_{min}$ packets of the preamble, but OPT is only able to complete at most as much work, so $C_O^1 \leq C_A^1$. For phases of type 2, we observe that the amount of work completed by OPT minus the work completed by SL-Preamble is at most ℓ_{max} (i.e., $C_O^2 - C_A^2 < \ell_{max}$). Therefore, $C_O^2 \leq \frac{\ell_{min}\overline{\gamma}}{\ell_{max}+\ell_{min}\overline{\gamma}} C_A^2$. (Observe that $\frac{\ell_{min}\overline{\gamma}}{\ell_{max}+\ell_{min}\overline{\gamma}} \leq 1/2$.) The same holds for phases of type 4 ($C_O^4 - C_A^4 < \ell_{max}$) and hence in this case $C_O^4 \leq 2C_A^4$. In the case of phases of type 3, SL-Preamble is not able to transmit successfully any packet, and therefore $C_A^3 = 0$, whereas OPT might transmit up to $\hat{\gamma}\ell_{min}$ packets.

There are two cases of executions to be considered separately.

Case 1: The number of phases of type 3 is finite.

In such a case, there is a phase i^* such that $\forall i > i^*$ phase i is not of type 3. Then

$$R_1 = \frac{\sum\limits_{j\leq i^*} C_A(j) + \sum\limits_{j>i^*} C_A(j)}{\sum\limits_{j\leq i^*} C_O(j) + \sum\limits_{j>i^*} C_O(j)} \tag{2}$$

It is clear that the total progress completed by the end of phase i^* by both algorithms is bounded. So we define $\sum\limits_{j\leq i^*} C_A(j) = A$ and $\sum\limits_{j\leq i^*} C_O(j) = O$ and thus,

$$R_1 = \frac{A + \sum\limits_{j>i^*} C_A(j)}{O + \sum\limits_{j>i^*} C_O(j)} \geq \frac{A + \frac{\ell_{min}\overline{\gamma}}{\ell_{max}+\ell_{min}\overline{\gamma}} \sum\limits_{j>i^*} C_O(j)}{O + \sum\limits_{j>i^*} C_O(j)}$$

Hence, the relative throughput of SL-Preamble at the end of each phase, can be computed as $T = \lim_{t\to\infty} R_1$, i.e.,

$$\begin{aligned}
T &= \lim_{j\to\infty} \frac{A + \frac{\ell_{min}\overline{\gamma}}{\ell_{max}+\ell_{min}\overline{\gamma}} \sum\limits_{j>i^*} C_O(j)}{O + \sum\limits_{j>i^*} C_O(j)} \\
&= \lim_{j\to\infty} \frac{(\ell_{max}+\ell_{min}\overline{\gamma})A + (\ell_{min}\overline{\gamma}) \sum\limits_{j>i^*} C_O(j)}{(\ell_{max}+\ell_{min}\overline{\gamma})(O + \sum\limits_{j>i^*} C_O(j))} \\
&= \lim_{j\to\infty} \left(\frac{\ell_{min}\overline{\gamma}}{\ell_{max}+\ell_{min}\overline{\gamma}} + \frac{(\ell_{max}+\ell_{min}\overline{\gamma})A - (\ell_{min}\overline{\gamma})O}{(\ell_{max}+\ell_{min}\overline{\gamma})(O + \sum\limits_{j>i^*} C_O(j))}\right) \\
&= \frac{\ell_{min}\overline{\gamma}}{\ell_{max}+\ell_{min}\overline{\gamma}} = \frac{\overline{\gamma}}{\gamma+\overline{\gamma}}
\end{aligned}$$

Here it is important to note that the assumption $\lim_{t\to\infty} C_O(t) = \infty$ is used, which corresponds to the expression $\lim_{j\to\infty} \sum\limits_{j>i^*} C_O(j)$ in the above equality.

So far, we have basically seen what is the relative throughput of SL-Preamble at the end of each phase. It is also important to guarantee the lower bound at all times within the phases. Consider any time-point t of phase $i > i^*$. Then $R_i(t) = \frac{\sum_{j\in(i^*,i-1]} C_A(j)+X_t}{\sum_{j\in(i^*,i-1]} C_O(j)+Y_t}$, where X_t and Y_t is the work completed by SL-Preamble and OPT within phase i up to time t. Using our proof above and the fact that for phases of type 1, 2 and 4 $C_A \geq \frac{\ell_{min}\overline{\gamma}}{\ell_{max}+\ell_{min}\overline{\gamma}} C_O$, we know that $X_t \geq \frac{\ell_{min}\overline{\gamma}}{\ell_{max}+\ell_{min}\overline{\gamma}} Y_t$ as well. Therefore,

$$\begin{aligned} R_i(t) &\geq \frac{\frac{\ell_{min}\overline{\gamma}}{\ell_{max}+\ell_{min}\overline{\gamma}} \sum_{j\in(i^*,i-1]} C_O(j) + \frac{\ell_{min}\overline{\gamma}}{\ell_{max}+\ell_{min}\overline{\gamma}} Y_t}{\sum_{j\in(i^*,i-1]} C_O(j) + Y_t} \\ &= \frac{\ell_{min}\overline{\gamma}}{\ell_{max}+\ell_{min}\overline{\gamma}} \end{aligned}$$

This completes the lower bound of relative throughput for Case 1.

Case 2: The number of phases of type 3 is infinite.
In this case we must see how the number of ℓ_{min} and ℓ_{max} packets are bounded for both SL-Preamble and OPT.

Lemma 1 *Consider the time point t at the beginning of a phase j of type 3. Then the number of ℓ_{min} tasks completed by t by OPT is no more than the amount of ℓ_{min} tasks completed by SL-Preamble plus $\overline{\gamma}-1$, i.e., $\sum_{i<j} a_i^* \leq \sum_{i<j}(a_i + c_i) + (\overline{\gamma}-1)$.*

Proof: Consider the beginning of phase j of type 3. At that point, we know that SL-Preamble has at most $(\overline{\gamma}-1)\ \ell_{min}$ tasks in its queue of pending tasks by definition of phase type 3. Therefore, the amount of ℓ_{min} tasks completed by OPT by the beginning of phase j is no more than the ones completed by SL-Preamble (including the ℓ_{min} tasks in preambles) plus $\overline{\gamma}-1$. ■

Lemma 2 *Considering all kinds of phases and the number of ℓ_{max} tasks, $\sum_{i\leq j} b_i^* \leq \sum_{i\leq j} b_i + \sum_{i\leq j} \frac{c_i}{\overline{\gamma}} + 2, \forall j$*

Proof: We prove this claim by induction on phase j. For the *Base Case:* $j = 0$ the claim is trivial. We consider the *Induction Hypothesis* stating that $\sum_{i\leq j-1} b_i^* \leq \sum_{i\leq j-1} b_i + \sum_{i\leq j-1} \frac{c_i}{\overline{\gamma}} + 2$. For the *Induction Step* we need to prove it up to the end of phase j. We first consider the case where during the phase j there is a time when SL-Preamble has no ℓ_{max} tasks. Let t be the latest such time in the phase. Let us define $b^*(t)$ and $b(t)$ being the number of ℓ_{max} tasks completed up to time t by OPT and SL-Preamble respectively. We know that $b^*(t) \leq b(t)$. Let also $x_j^*(t)$ and $x_j(t)$ be the number of ℓ_{max} tasks completed by OPT and SL-Preamble, respectively, after time point t until the end of the phase j. We claim that $x_j^*(t) \leq x_j(t) + 2$. From our definitions, at time t SL-Preamble is executing a ℓ_{min} task. Since t is the last time that SL-Preamble has no ℓ_{max} tasks, the worst case is being at the beginning of the preamble (by inspection of the 4 types of phases). Then, if the phase ends at time t', we define period $I = [t, t']$:

$$\begin{aligned} |I| &< \overline{\gamma}\ell_{min} + (x_j(t)+1)\ell_{max} \\ &\leq (x_j(t)+2)\ell_{max} \end{aligned}$$

The +1 ℓ_{max} task is because of the crash before completing the last ℓ_{max} scheduled task of the phase. Observe that OPT could be executing a ℓ_{max} task at time t, completed at some point in $[t, t+\ell_{max}]$ and accounted for in $x_j^*(t)$. Therefore,

$$\sum_{i\leq j} b_i^* = b^*(t) + x_j^*(t) \leq b(t) + x_j(t) + 2 = \sum_{i\leq j} b_i + 2.$$

Now consider the case where at all times of a phase j there are ℓ_{max} tasks in the queue of SL-Preamble. By inspection of the 4 types of phases, the worst case is when j is of type 2. Since there is always some ℓ_{max} task pending in SL-Preamble, after completing the $\overline{\gamma}\ell_{min}$ tasks it will keep scheduling ℓ_{max} tasks, until a crash stops the last one scheduled, or the queue becomes empty. On the same time OPT is able to complete at most $\lfloor \frac{L_j}{\ell_{max}} \rfloor \leq b_j + 1$ ℓ_{max}-tasks, where L_j is the length of the phase. Therefore, in all types of phases, $b_j^* \leq \frac{c_j}{\overline{\gamma}} + b_j$. And hence by induction the claim follows; $\sum\limits_{i \leq j} b_i^* \leq \sum\limits_{i \leq j} \frac{c_i}{\overline{\gamma}} + \sum\limits_{i \leq j} b_i + 2$. ■

Combining the two lemmas above, Lemma 1 and 2:

$$R_2 = \frac{\sum\limits_{i \leq j} C_A(i)}{\sum\limits_{i \leq j} C_O(j)} = \frac{\sum\limits_{i \leq j} [(a_i + c_i)\ell_{min} + b_i \ell_{max}]}{\sum\limits_{i \leq j} [a_i^* \ell_{min} + b_i^* \ell_{max}]}$$

$$\begin{aligned}
&\geq \frac{\sum\limits_{i \leq j} [(a_i + c_i)\ell_{min} + b_i \ell_{max}]}{\sum\limits_{i \leq j} (a_i + c_i)\ell_{min} + (\overline{\gamma} - 1)\ell_{min} + \sum\limits_{i \leq j} (b_i + \frac{c_i}{\overline{\gamma}})\ell_{max} + 2\ell_{max}} \\
&\geq \frac{\sum\limits_{i \leq j} [(a_i + c_i)\ell_{min} + b_i \ell_{max}]}{\sum\limits_{i \leq j} [(a_i + 2c_i)\ell_{min} + b_i \ell_{max}] + 3\ell_{max}} \\
&\geq \frac{\sum\limits_{i \leq j} [(a_i + c_i)\ell_{min} + b_i \ell_{max}] + \frac{3}{2}\ell_{max} - \frac{3}{2}\ell_{max}}{2\sum\limits_{i \leq j} [(a_i + c_i)\ell_{min} + b_i \ell_{max}] + 3\ell_{max}} \\
&\geq \frac{1}{2} - \frac{\frac{3}{2}\ell_{max}}{2\sum\limits_{i \leq j} [(a_i + c_i)\ell_{min} + b_i \ell_{max}] + 3\ell_{max}}
\end{aligned}$$

Therefore,

$$T = \lim_{j \to \infty} R_2 \geq \frac{1}{2} \tag{3}$$

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

Proof: From the analyses of Cases 1 and 2 and the fact that $\frac{\overline{\gamma}}{\gamma + \overline{\gamma}} \leq \frac{1}{2}$ it is easy to conclude that the relative throughput of Algorithm SL-Preamble is at least $\frac{\overline{\gamma}}{\gamma + \overline{\gamma}}$ as claimed. ■

4 Stochastic Arrivals

We now turn our attention to stochastic packet arrivals.

4.1 Upper Bounds

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 these 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 $j \leq \hat{\gamma}$ packets of length ℓ_{min} are sent by OFF completed by time t. The value $r_{2a}^{(j)}$ with $j \leq \overline{\gamma}$ packets of length ℓ_{min} 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

$$T_{\text{Alg}}(A, E, t) = \frac{\ell_{min} \sum_{j=1}^{\overline{\gamma}} j \cdot r_{2a}^{(j)}}{\ell_{max} \sum_{j=1}^{\overline{\gamma}} r_{2a}^{(j)} + \ell_{min} \sum_{j=1}^{\hat{\gamma}} j \cdot r_{1a}^{(j)}} \,. \tag{4}$$

From this expression, we can show the following result.

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

Proof: It can be observed in Eq. 4 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 3 *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 $1 - e^{-c't}$.*

Proof: We first prove the statement 1(a). The Poisson process governing arrival times of packets of length ℓ_{min} has parameter λp. By the definition of a Poisson process, the distribution of packets of length ℓ_{min} arriving to the system in the period $[0, t]$ is the Poisson distribution with parameter λpt. Consequently, by Chernoff bound for Poisson random variables (with parameter λpt), c.f., [8], the probability that at least ηpt packets arrive to the system in the period $[0, t]$ is at least

$$1 - e^{-\lambda pt} \frac{(e\lambda pt)^{\eta pt}}{(\eta pt)^{\eta pt}} = 1 - e^{-tp(\lambda - \eta \ln(e\lambda/\eta))} \geq 1 - e^{-ct},$$

for some constant $c > 0$ dependent on λ, η, p. In the above, the argument behind the last inequality is as follows. It is a well-known fact that $x > 1 + \ln x$ holds for any $x > 1$; in particular, for $x = \lambda/\eta > 1$. This implies that $x - \ln(ex)$ is a positive constant for $x = \lambda/\eta > 1$, and after multiplying it by $\eta > 0$ we obtain another positive constant equal to $\lambda - \eta \ln(e\lambda/\eta)$ that depends only on λ and η. Finally, we multiply this constant by $p > 0$ to obtain the final constant $c > 0$ dependent only on λ, η, p.

The same result for packets of length ℓ_{max} can be proved by replacing p by $q = 1 - p$ in the above analysis.

Statement 1(b) is proved analogously to the first one, by replacing η by η'. This is possible because the Chernoff bound for Poisson process has the same form regardless whether the upper or the lower bound on the Poisson value is considered, c.f., [8]. ■

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 no

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

$$T_{\text{Alg}}(A,E,t_i) \leq \frac{s_{t_i}^{alg}\ell_{min}}{r_{t_i}^{off}\ell_{max}+a_{t_i}^{\min}\ell_{min}} \leq \frac{s_{t_i}^{alg}\ell_{min}}{\lfloor\frac{s_{t_i}^{alg}}{\overline{\gamma}}\rfloor\ell_{max}+a_{t_i}^{\min}\ell_{min}} \leq \frac{s_{t_i}^{alg}\ell_{min}}{(\frac{s_{t_i}^{alg}}{\overline{\gamma}}-1)\ell_{max}+a_{t_i}^{\min}\ell_{min}}.$$

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 3, we have that the number of ℓ_{min} packets injected by time t satisfy $a_t^{\min} > \eta' pt$ with probability at most $\exp(-c't)$ and the injected $\max$ packets satisfy $a_t^{\max} < \eta qt$ 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 3, we can also claim that in the interval $(t_0, t_i]$ the probability that more than $\eta' pi\delta$ 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.

Then we show the following:

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: We consider three complementary cases.

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 $\frac{\overline{\gamma}}{2\gamma} \leq \lambda p \ell_{min} \leq 1$**.** Our goal is to prove that the relative throughput is not smaller than $\min\left\{\eta p \ell_{min}, \frac{\overline{\gamma}}{\gamma}\right\}$, for any η satisfying $\lambda/2 < \eta < \lambda$. Considering such an η we can make use of Lemma 3 with respect to λ, η, p. The relative throughput compares the behavior of algorithm CSL-Preamble, which is simply SL in this case, with OPT for each execution. Hence, for the purpose of the analysis we introduce the following modification in every execution: we remove all periods in which OPT is not transmitting any packet. By "removing" we understand that we count time after removing the OPT-unproductive periods and "gluing" the remaining periods so that they form one time line. In the remainder of the analysis of this case we consider these modified executions with modified time lines and whenever we need to refer to the "original" time line we use the notion of *global time*.

For any positive integer i, we define time points $t_i = i \cdot \ell_{max}$. Consider events S_i, for positive integers i, defined as follows: the number of packets arrived by time t_i (on the modified time line of the considered execution) is at least $t_i \eta p$. By Lemma 3 and the fact that time t on the modified time line cannot occur before the global time t, there is a constant c dependent only on λ, η, p such that for any i: the event S_i holds with probability at least $1 - \exp(-ct_i)$.

Consider an integer $j > 1$ being a square of another integer. We prove that by time t_j, the relative throughput 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 $1 - c' \exp(-ct_{\sqrt{j}})$, for some constant $c' > 1$ dependent only on λ, η, p. To show this, consider two complementary scenarios that may happen at time t_j: there are at least $\overline{\gamma}$ pending packets of length ℓ_{min}, or otherwise. It is sufficient to show the sought property separately in each of these two scenarios.

Consider the first scenario, when there are at least $\overline{\gamma}$ pending packets of length ℓ_{min} at time t_j. With probability at least $1 - c' \exp(-ct_{\sqrt{j}})$, for every $\sqrt{j} \leq i \leq j$ at least $t_i \eta p$ packets arrive by time t_i. This is because of the union bound of the corresponding events S_i and the fact that $\sum_{i \geq \sqrt{j}} \exp(-ct_i) \leq c' \cdot \exp(-ct_{\sqrt{j}})$ for some constant $c' > 1$ dependent on λ, η, p (note here that although c' seems to depend also on c, c' is still dependent only on λ, η, p because c is a function of these three parameters as well). Consider executions in $\bigcup_{i=\sqrt{j}}^{j} S_i$. Using induction on i, if follows that for these executions for every $\sqrt{j} \leq i \leq j$ the following invariant holds: at least $t_i \eta p - \overline{\gamma}$ packets of length ℓ_{min} have been successfully transmitted by time t_i or in the time interval $[t_i, t_{i+1}]$ at least $\overline{\gamma}$ packets of length ℓ_{min} are successfully transmitted (i.e., these successful transmissions end in the interval $[t_i, t_{i+1}]$). The inductive proof of this invariant follows directly from the specification of algorithm CSL-Preamble (recall that it simply runs algorithm SL in the currently considered case) and from the definition of the modified execution and time line. Let i^* denote the largest $i \in [\sqrt{j}, j]$ satisfying the following condition: there are less than $\overline{\gamma}$ packets of length ℓ_{min} pending in time t_i; if such an i does not exist, we set $i^* = -1$. Consider two sub-cases.

Sub-case $i^* \geq \sqrt{j}$. If follows from the invariant and the definition of i^* that by time t_{i^*} there are at least $t_i \eta p - \overline{\gamma}$ successfully transmitted packets of length ℓ_{min}, and in each interval $[t_i, t_{i+1}]$, for $i^* \leq i < j$, at least $\overline{\gamma}$ packets of length ℓ_{min} finish their successful transmission. Therefore, by time t_j the total length of packets (of length ℓ_{min}) successfully transmitted by algorithm CSL-Preamble is at least

$$(t_{i^*} \eta p - \overline{\gamma})\ell_{min} + \frac{t_j - t_{i^*}}{\ell_{max}} \cdot \overline{\gamma}\ell_{min} \ ,$$

while the total length of successfully transmitted packets by OPT by time t_j is at most t_j, by the definition of the modified execution and time line. Therefore the relative throughput is at least

$$\frac{(t_{i^*} \eta p - \overline{\gamma})\ell_{min} + \frac{t_j - t_{i^*}}{\ell_{max}} \cdot \overline{\gamma}\ell_{min}}{t_j}$$

$$\geq \min\left\{\frac{(t_j\eta p-\overline{\gamma})\ell_{min}}{t_j}, \frac{\frac{t_j-t_{\sqrt{j}}}{\ell_{max}}\cdot\overline{\gamma}\ell_{min}}{t_j}\right\}$$

$$= \min\left\{\eta p\ell_{min}-\frac{\overline{\gamma}\ell_{min}}{t_j}, (1-1/\sqrt{j})\cdot\frac{\overline{\gamma}}{\gamma}\right\} .$$

This converges to $\min\left\{\eta p\ell_{min}, \frac{\overline{\gamma}}{\gamma}\right\}$ with j going to infinity.

Sub-case $i^* < \sqrt{j}$. In this sub-case we have, by definition of $i^* < \sqrt{j}$, that at every time t_i, where $\sqrt{j} \leq i \leq j$, there are at least $\overline{\gamma}$ pending packets of length ℓ_{min}. Consequently, by the specification of the algorithm, in each interval $[t_i, t_{i+1}]$, for $\sqrt{j} \leq i < j$, at least $\overline{\gamma}$ packets of length ℓ_{min} finish their successful transmission. Therefore, by time t_j the total length of packets (of length ℓ_{min}) successfully transmitted by algorithm CSL-Preamble is at least

$$\frac{t_j - t_{\sqrt{j}}}{\ell_{max}}\cdot\overline{\gamma}\ell_{min} ,$$

while the total length of successfully transmitted packets by OPT by time t_j is at most t_j, by the definition of the modified execution and time line. Therefore the relative throughput is at least

$$\frac{\frac{t_j-t_{\sqrt{j}}}{\ell_{max}}\cdot\overline{\gamma}\ell_{min}}{t_j} = (1-1/\sqrt{j})\cdot\frac{\overline{\gamma}}{\gamma} ,$$

and it converges to $\frac{\overline{\gamma}}{\gamma}$ with j going to infinity. This completes the analysis of the sub-cases.

Finally, it is important to notice that the final converge of the ratio, with j going to infinity, in both sub-cases gives a valid bound on the relative throughput, since the subsequent ratios hold with probabilities approaching 1 exponentially fast (in j), i.e., with probabilities at least $1 - c' \exp(-ct_{\sqrt{j}})$, where c and c' are positive constants dependent only on λ, η, p. The minimum of the two relative throughputs, coming from the sub-cases, is $\min\left\{\eta p\ell_{min}, \frac{\overline{\gamma}}{\gamma}\right\}$, as desired and therefore the relative throughput is at least $\min\left\{\lambda p\ell_{min}, \frac{\overline{\gamma}}{\gamma}\right\}$ in this case.

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 $\min\left\{\eta p\ell_{min}, \frac{\overline{\gamma}}{\gamma}\right\}$, for any $\lambda/2 < \eta < \lambda$, and therefore it is at least $\min\left\{\lambda p\ell_{min}, \frac{\overline{\gamma}}{\gamma}\right\} \geq \min\left\{1, \frac{\overline{\gamma}}{\gamma}\right\} = \frac{\overline{\gamma}}{\gamma}$. ■

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 was motivated by the following observation regarding the system of dynamic packet arrivals with errors: scheduling packets of same length is relatively easy and efficient in case of instantaneous feedback, but extremely inefficient in case of deferred feedback. 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 would be 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] 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.

[4] 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.

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

[6] Shu Lin and Daniel J Costello. *Error control coding*, volume 123. Prentice-hall Englewood Cliffs, NJ, 2004.

[7] Chad Meiners and Eric Torng. Mixed criteria packet scheduling. *Algorithmic Aspects in Information and Management*, pages 120–133, 2007.

[8] Michael Mitzenmacher and Eli Upfal. *Probability and Computing*. Cambridge University Press, 2005.

[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 multihop 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.

APPENDIX

A NP-hardness

We prove the NP-hardness of the following problem, defined for a single link.

INSTANCE (Throughput Problem): Set X of packets, for each packet $x \in X$ a length $l(x) \in \mathbb{N}^+$, an arrival time $a(x) \in \mathbb{Z}^0$, a sequence of time instants $0 = T_0 < T_1 < T_2 < \cdots < T_k$, $T_i \in \mathbb{N}^0$, so that the link suffers an instantaneous error at each time T_i, $i \in [1, k]$ (in other words, at each time T_i, any packet transmitted over the link is corrupted).

QUESTION: is there a schedule of X so that error-free packets of total length T_k are transmitted by time T_k over the link?

Theorem 6 *The Throughput Problem is NP-hard.*

Proof: We use the 3-Partition problem which is known to be an NP-hard problem.

INSTANCE: Set A of $3m$ elements, a bound $B \in \mathbb{N}^+$ and, for each $a \in A$, a size $s(a) \in \mathbb{N}^+$ such that $B/4 < s(a) < B/2$ and $\sum_{a \in A} s(a) = mB$.

QUESTION: can A be partitioned into m disjoint sets $\{A_1, A_2, \ldots, A_m\}$ such that, for each $1 \leq i \leq m$, $\sum_{a \in A_i} s(a) = B$?

We reduce the 3-Partition problem to the Throughput Problem, defined for a single link. The reduction is by setting $X = A$, $l() = s()$, $a() = 0$, $k = m$, and $T_i = iB$ for $i \in [1, k]$. If the answer to 3-Partition is affirmative, then for the Throughput Problem there is a way to schedule (and transmit) the packets in X in subsets $\{X_1, X_2, \ldots, X_m\} = \{A_1, A_2, \ldots, A_m\}$, so that all the packets in A_i can be transmitted over the link in the interval $[T_{i-1}, T_i]$. Furthermore, since $\sum_{a \in A_i} s(a) = \sum_{x \in X_i} l(x) = B$, and $T_i - T_{i-1} = B$, the total length of packets transmitted by time T_k is T_k.

The reverse argument is similar. If there is a way to schedule packets so that the total packet length transmitted by time T_k is T_k, in each interval between two error events on the link there must be exactly B bytes of packets transmitted. Then, the packets can be partitioned into subsets of total length B each. This implies the partition of A. ■

B Deferred Feedback

In this section we study the relative throughput of any algorithm under the deferred feedback mechanism. As described in Section 1, with this mechanism the sending station is notified about a packet having been corrupted by an error only after the transmission of the packet is completed. Here we assume that all packets have the same length ℓ. We show that even in this case no algorithm can achieve positive throughput.

B.1 Adversarial Arrivals

In order to prove the upper bound on throughput, the packets arrive frequently enough so that there are always packets ready. The algorithm will then greedily send a train of packets. The adversary injects bit errors at a distance of exactly ℓ so that each error hits a different packet, and hence the algorithm cannot successfully complete any transmission (that is, it cannot transmit non-corrupted packets). At the same time, an offline algorithm OFF is able to send packets in each interval of length ℓ without errors. This argument leads to the following theorem:

Theorem 7 *No packet scheduling algorithm Alg can achieve a relative throughput larger than* 0 *under adversarial arrivals in the deferred feedback model, even with one packet length.*

B.2 Stochastic Arrivals

Let us consider now stochastic arrivals. We show that also in this case the upper bound on the relative throughput is 0.

Theorem 8 *No packet scheduling algorithm Alg can achieve a relative throughput larger than* 0 *under stochastic arrivals in the deferred feedback model, even with one packet length.*

Proof: As described in Section 2, we assume that packets arrive at a rate λ. Here we assume that all packets have the same length ℓ. Observe that if $\lambda\ell < 1$ there are many times when there is no packet ready to be sent and the link will be idle. In any case, the adversary can inject errors following the next rule: inject an error in the middle point of each packet sent by Alg. Applying this rule, no packet sent by Alg is received without errors. However, between two errors there is at least ℓ space (even if packets are contiguous) and the offline algorithm OFF can send a packet. The conclusion is that OFF is able to successfully send at least one packet between two attempts of Alg, while Alg cannot complete successfully any transmission. This completes the proof. ■

C Upper Bounds for Algorithms SL and LL

We prove upper bounds that suggest that algorithms SL (Shortest Length) and LL (Longest Length) are not efficient. First, we show that SL cannot have relative throughput larger than $\frac{1}{\gamma+1}$ under adversarial arrivals. We then show that algorithm LL is even worse, as its relative throughput cannot be more than 0 even with stochastic arrivals.

Theorem 9 *Algorithm SL cannot achieve relative throughput larger than* $\frac{1}{\gamma+1}$ *under adversarial arrivals, even if there is a schedule that transmits all the packets.*

Proof: The scenario works as follows. At time 0 two packets arrive, one of length ℓ_{max} and one of length ℓ_{min}. SL schedules first the packet of length ℓ_{min}, and when it is transmitted, it schedules the packet of length ℓ_{max}. Meanwhile, an offline algorithm OFF schedules first the packet of length ℓ_{max}. When it is transmitted, the adversary causes an error on the link, so SL does not transmit successfully the packet of length ℓ_{max}. Now, SL only has one packet of length ℓ_{max} in its queue (when this scenario is repeated will have several, but no packets of length ℓ_{min}). Hence, SL schedules this packet, while OFF schedules the packet of length ℓ_{min} that has in its queue. When OFF completes the transmission of the ℓ_{min} packet, the adversary causes an error on the link. This scenario can be repeated forever. In each instance, OFF transmits one packet of length ℓ_{max} and one of lenght ℓ_{min}, while SL only transmits one packet of length ℓ_{min}. Hence, the throughput achieved is $\frac{\ell_{min}}{\ell_{max}+\ell_{min}} = \frac{1}{\gamma+1}$. Observe that at the end of each instance of the scenario the queue of OFF is empty. ■

We now show that the above upper bound also holds with stochastic arrivals under specific packet arrival rates.

Theorem 10 $\forall\varepsilon > 0, \exists\lambda, p, q$ *such that algorithm SL cannot achieve a relative throughput larger than* $\frac{1}{(1-\varepsilon)\gamma+1} + \varepsilon$.

Proof: Consider an execution of the SL algorithm. We define intervals $I_1, I_2, \ldots, I_i$ as follows. The first such interval, I_1, starts with the arrival of the first ℓ_{min} packet. Then, I_i starts as soon as an ℓ_{min} packet is in the queue of SL after the end of interval I_{i-1}. The length of each interval depends on whether OFF has an

ℓ_{max} packet in its queue at the start of the interval or not. If it has an ℓ_{max} packet, the length of the interval is $|I_i| = \ell_{min} + \ell_{max}$, and we say that we have a *long* interval. If it does not, the length is $|I_i| = \ell_{min}$ and the interval is called *short*.

Between intervals the adversary injects frequent errors, so SL cannot transmit any packet. In every interval I_i, SL starts by scheduling an ℓ_{min} packet. In a short interval, OFF sends an ℓ_{min} packet, followed by an error injected by the adversary. Hence, in a short interval both SL and OFF successfully transmit one ℓ_{min} packet. In a long interval, OFF sends an ℓ_{max} packet, after which the adversary injects an error. (Up to that point SL has been able to complete the transmission of one or more ℓ_{min} packets, but no ℓ_{max} packet.) After the error, OFF sends an ℓ_{min} packet (which is available since beginning of the interval) after which continuous errors will be injected by the adversary until the next interval. Hence, in a long interval OFF successfully transmits one ℓ_{min} packet and one ℓ_{max} packet, while SL transmits only ℓ_{min} packets. This implies that in both types of intervals OFF is transmitting useful packets during the whole interval.

Let us denote by s_k the total length of the intervals $I_1, I_2, \ldots, I_k$, i.e., $s_k = \sum_{i=1}^{k} |I_i|$. Observe that the total number of ℓ_{min} packets that arrive up to the end of interval I_k is bounded by k (that accounts for the ℓ_{min} packet in the queue of SL at the start of each interval) plus the packets that arrive in the intervals. From Lemma 3, we know that there is a constant $\eta' > \lambda$ and a constant $c' > 0$ which depends only on η', λ and p, such that the number of ℓ_{min} packets that arrive in the intervals is at most $\eta' p s_k$ with probability at least $1 - e^{-c' s_k}$.

Let T_k be the throughput of SL at the end of interval I_k. From the above, we have that T_k is bounded as

$$T_k \leq \frac{\ell_{min}(k + \eta' p s_k)}{s_k} = \frac{\ell_{min} k}{s_k} + \ell_{min} \eta' p$$

with probability at least $\pi_1(k) = 1 - e^{-c' s_k}$. Observe that in the above expression it is assumed that all ℓ_{min} packets that arrive by the end of I_k are successfully transmitted by SL. We provide now the following claim.

Claim: Let us consider the first $x + 1$ intervals I_i, for $x > 1$. The number of long intervals is at least $(1 - \delta)(1 - e^{-\lambda q \ell_{min}})x$ with probability at least $1 - \exp(-\delta^2 (1 - e^{-\lambda q \ell_{min}})x/2)$, for any $\delta \in (0, 1)$.
Proof of claim: Observe that if an ℓ_{max} packet arrives during interval I_i then the next interval I_{i+1} is long. We consider now the first x intervals. Since each of these intervals has length at least ℓ_{min}, some ℓ_{max} packet arrives in the interval with probability at least $1 - e^{-\lambda q \ell_{min}}$ (independently of what happens in other intervals). Hence, using a Chernoff bound, the probability of having less than $(1 - \delta)(1 - e^{-\lambda q \ell_{min}})x$ intervals among the x first intervals in which ℓ_{max} packets arrive is at most $\exp(-\delta^2 (1 - e^{-\lambda q \ell_{min}})x/2)$. □

From the claim, it follows that there are at least $(1 - \delta)(1 - e^{-\lambda q \ell_{min}})(k - 1)$ long intervals among the first k intervals, with high probability. Hence, the value of s_k is bounded as

$$\begin{aligned} s_k &\geq (1 - \delta)(1 - e^{-\lambda q \ell_{min}})(k - 1)(\ell_{max} + \ell_{min}) + (k - (1 - \delta)(1 - e^{-\lambda q \ell_{min}})(k - 1))\ell_{min} \\ &= (1 - \delta)(1 - e^{-\lambda q \ell_{min}})(k - 1)\ell_{max} + k\ell_{min} \end{aligned}$$

with probability at least $\pi_2(k) = 1 - \exp(-\delta^2 (1 - e^{-\lambda q \ell_{min}})(k - 1)/2)$. Note that T_K cannot be larger than 1. Hence, the expected value of T_k can be bounded as follows.

$$\mathbb{E}[T_k] \leq \pi_1(k)\pi_2(k) \left(\frac{\ell_{min} k}{(1 - \delta)(1 - e^{-\lambda q \ell_{min}})(k - 1)\ell_{max} + k\ell_{min}} + \ell_{min} \eta' p \right) + (1 - \pi_1(k)\pi_2(k)).$$

Since $\pi_1(k)$ and $\pi_2(k)$ tend to one as k tends to infinity, we have that

$$\begin{aligned}\lim_{k\to\infty} \mathbb{E}[T_k] &\leq \frac{\ell_{min}}{(1-\delta)(1-e^{-\lambda q \ell_{min}})\ell_{max} + \ell_{min}} + \ell_{min}\eta' p \\ &= \frac{1}{(1-\delta)(1-e^{-\lambda q \ell_{min}})\gamma + 1} + \ell_{min}\eta' p.\end{aligned}$$

Hence, choosing η', p, q, and δ appropriately, the claim of the theorem follows. (E.g., they must satisfy $\ell_{min}\eta' p \leq \varepsilon$ and $(1-\delta)(1-e^{-\lambda q \ell_{min}}) \geq (1-\varepsilon)$.) ■

Theorem 11 *Algorithm LL cannot achieve relative throughput larger than* 0*, even under stochastic arrivals.*

Proof: The scenario is simple. The adversary blocks all successful transmissions (by placing errors at distance smaller than ℓ_{min}) until at least two packets have arrived, one of length ℓ_{max} and one of length ℓ_{min}. Algorithm LL schedules a packet of length ℓ_{max}, while an offline algorithm OFF schedules a packet of length ℓ_{min}. Once OFF completes the transmission of this packet, the adversary causes an error on the link, and hence LL does not complete the transmission of the ℓ_{max} packet. Then, again the adversary blocks successful transmissions until OFF has at least one ℓ_{min} packet pending. The scenario is repeated for ever; while OFF will be transmitting successfully all ℓ_{min} packets, LL will be stuck on the unsuccessful transmissions of ℓ_{max} packets. Hence, the throughput will be 0. ■

D Randomized Algorithms

So far we have considered deterministic solutions. In many cases, randomized solutions can obtain better performance. As we argue in this section, this is not the case for the problem considered in this work.

Let us first indicate how the model and the definition of relative throughput must be extended to the case of randomized algorithms. We assume that the adversary knows the algorithm and the history of the random choices made by the algorithm until the current point in time, but it does not know the future random choices made by the algorithm.

Regarding the relative throughput, and following the terminology of Section 2, in the case of randomized algorithms, an adversarial error-function E has three arguments: an arrival pattern A, a string of values of random bits R, and time t. The output of $E(A, R, t)$ is a set of errors until time t based on the execution of a given randomized algorithm with the values of random bits taken from R under an adversarial pattern A by round t.

For arrival pattern A, adversarial error-function E, string of random bits R and time t, we define the *relative throughput* $T_{Alg}(A, E, R, t)$ *of a randomized algorithm Alg by time* t as follows:

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

$T_{\text{Alg}}(A, E, R, t)$ is defined as 1 if $L_{\text{Alg}}(A, E, R, t) = L_{\text{OPT}}(A, E, R, t) = 0$.

(Note that OPT is not randomized, but since the error-function E depends on the random choices of the algorithm, this has a direct effect on the performance of OPT.)

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

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

where A is understood as a function of R and t, and $\mathcal{R}$ is a distribution of all possible strings of random bits used by the algorithm. In the stochastic arrival model the relative throughput needs to take into account the random distribution of arrival patterns in $\mathcal{A}$ (they are not functions now, as they do not depend on the adversary), and it is defined as follows:

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

Now, looking at the analyses of the upper bounds for deterministic algorithms with deferred feedback (Section B) and with instantaneous feedback (under adversarial arrivals, Section 3.1, and stochastic arrivals, Section 4.1), it is not difficult to see that the derived bounds hold also for randomized algorithms. The main observation that leads to this conclusion is the following: The adversarial error and arrival patterns defined in the analyses are reactive, in the sense that the adversary that controls them does not need to know the future (and in particular the future random bits of the algorithm) and makes its decisions only by looking at the system's history. In other words, when a given algorithm decides in a given phase what packet length to transmit, the adversary reacts adaptively on the specific choice, regardless of whether this choice was done deterministically or by flipping a coin. This leads to the conclusion that randomized solutions cannot yield better results (wrt relative throughput) for the considered packet scheduling problem.