J Sched
DOI 10.1007/s10951-015-0451-z

Measuring the impact of adversarial errors on packet scheduling strategies

Antonio Fernández Anta[1] · **Chryssis Georgiou**[2] · **Dariusz R. Kowalski**[3] · **Joeg Widmer**[1] · **Elli Zavou**[4]

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 to measure the efficiency of scheduling strategies in such a setting. To this end, we propose an *asymptotic 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 with deferred error feedback, which 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 asymptotic throughput achieved by these algorithms is shown to be close to optimal by deriving lower bounds on the metric 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.

Keywords Packet scheduling · Adversarial errors · Unreliable link · Asymptotic throughput · Competitive analysis · Error feedback mechanisms

1 Introduction

1.1 Motivation

Packet scheduling (Meiners and Torng 2007) is one of the fundamental problems in computer networks. As packets arrive, the sender (or scheduler) needs to continuously make scheduling decisions, without knowledge of future packet arrivals, and typically the objective is to maximize the *throughput* of the link or to achieve stability. Therefore, this problem is many times treated as an *online* scheduling problem (Awerbuch et al. 1992; Pruhs et al. 2004) and *competitive analysis* (Ajtai et al. 1994; Sleator and Tarjan 1985) 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 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*, considering the worst-case occurrence of errors; errors are caused by an omniscient and

✉ Elli Zavou
elli.zavou@imdea.org

1 IMDEA Networks Institute, Avda. del Mar Mediterraneo 22, 28918 Leganés, Madrid, Spain

2 Department of Computer Science, University of Cyprus, 75 Kallipoleos Str., P.O. Box 20537, 1678 Nicosia, Cyprus

3 Department of Computer Science, University of Liverpool, Ashton Building, Ashton Street, Liverpool L69 3BX, United Kingdom

4 Universidad Carlos III de Madrid and IMDEA Networks Institute, Avda. del Mar Mediterraneo 22, 28918 Leganés, Madrid, Spain

Published online: 30 September 2015

adaptive *adversary* (Richa et al. 2012). 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.

Specifically, we consider a single link between two stations, sender and receiver, with the sender scheduling the packets that arrive dynamically to be transmitted over the link. Packets may have lengths $\ell \in [\ell_{\min}, \ell_{\max}]$, where $\ell_{\min}$ and $\ell_{\max}$ are the smallest and largest lengths, respectively. We denote by $\gamma = \ell_{\max}/\ell_{\min}$, $\overline{\gamma} = \lfloor \gamma \rfloor$ and $\hat{\gamma} = \lceil \gamma \rceil - 1$. What is more, the arrival times are either controlled by the adversary, or are stochastic, following a Poisson distribution with parameter $\lambda > 0$. In this case, the packets have probability $p > 0$ of being of length $\ell_{\min}$ and probability $q > 0$ of being of length $\ell_{\max}$, where $p + q = 1$. However, the link is unreliable, that is, transmitted packets might be corrupted by bit errors. We consider an adversary controlling them, characterizing the worst-case scenarios, and look at *instantaneous* and *deferred* feedback mechanisms for the notification of the sender. For the performance evaluation, we pursue long-term competitive analysis. We denote by $T_{\text{Alg}} = \inf_{A\in\mathcal{A}, E\in\mathcal{E}} \lim_{t\to\infty} T_{\text{Alg}}(A, E, t)$ the asymptotic throughput of online algorithm Alg, where $\mathcal{A}$ is the set of packet arrival patterns and $\mathcal{E}$ the set of link error patterns. $T_{\text{Alg}}(A, E, t)$ is the throughput ratio of Alg under arrival and error patterns A and E up to time t.

1.2 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 (Andrews and Zhang 2005; Richa et al. 2012). Moreover, since a bit error renders a whole packet unusable (unless costly techniques like PPR (Jamieson and Balakrishnan 2007) 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.

– *Asymptotic throughput:* Our first contribution is the proposal of an *asymptotic throughput* metric for packet scheduling algorithms under unreliable links (Sect. 2). This metric is a variation of the competitive ratio typically considered in online scheduling and bin packing problems (see the works of Van Stee (2002) and Borodin and El-Yaniv (1998)). 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 are also controlled by an adversary, and (b) the arrival times and their sizes follow a stochastic distribution. We introduce this second model in order to decrease the power and control of the adversary over the system and compare with (a). In particular, we consider a Poisson distribution of arrivals (for details see Sect. 2), which is a classical distribution characterizing average cases. Note that it is an optimistic distribution for the realistic characterization of network traffic Becchi (2008). Nonetheless, we believe it of importance to analyze it before looking into other distributions, and as will be shown later, the analysis is not trivial. Note that the arrival model (a) gives results for the worst-case, while arrival model (b) gives results for the average-case. The general offline version of our scheduling problem, in which the scheduling algorithm knows a priori when errors will occur, is NP-hard (see Sect. 3). 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 (Lin and Costello 2004), 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. We show 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 turn our attention on 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 (Raghavan et al. 2001) that uses arithmetic coding to provide continuous error detection. The difference between deferred and instantaneous feedback is drastic, since for the instanta-

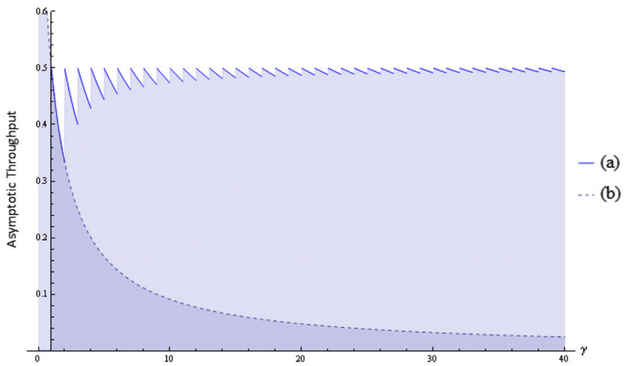


Fig. 1 Upper bound on the asymptotic throughput under adversarial packet arrivals and instantaneous feedback. **a** For any algorithm Alg, $T_{\text{Alg}} \leq \overline{\gamma}/(\gamma+\overline{\gamma})$. **b** For algorithm SL, $T_{\text{SL}} \leq 1/(\gamma+1)$. Observe that SL has a significantly lower bound as γ increases

neous feedback mechanism, and for packets of the same length, it is easy to obtain optimal asymptotic 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, $\ell_{\min}$ and $\ell_{\max}$, where $\ell_{\min} < \ell_{\max}$.

- *Bounds for adversarial arrivals:* We show (Sect. 4), that any online algorithm with instantaneous feedback can achieve at most almost half the asymptotic throughput with respect to the offline one. (See Fig. 1 for the graphical representation of the upper bound.) We also show 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 asymptotic 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 (Sect. 5), as one might expect, we obtain better asymptotic throughput in some cases. The results are summarized in Table 1 and a graphical representation can be seen better in Fig. 2. We propose and analyze an algorithm, called CSL-Preamble, that achieves optimal asymptotic throughput. It 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] of the asymptotic throughput for any algorithm in this setting.
- *A note on randomization:* All the proposed algorithms are deterministic. Using Yao's principle, Yao (1977), it follows that considering randomized algorithms does not improve the results; the upper bounds on the asymptotic throughput already discussed hold also in the randomized case, for oblivious adversaries (see Sect. 6).

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 asymptotic throughput under worst-case adversarial errors (almost half the asymptotic throughput that the offline optimal algorithm can achieve). Furthermore, we observe that in some cases, stochastic arrivals allow for better performance.

1.3 Related work

A vast amount of work exists for online scheduling. Here we focus only on the work that is most related to ours, but for more information we advice the reader to consult the works of Pinedo (2012) and Pruhs et al. (2004). The work of Kesselheim (2012) 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 *stability*, meaning bounded time from injection to delivery. The work of Andrews and Zhang (2005) 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 of Richa et al. (2012) 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. Another related work is the one of Anantharamu et al. (2011), in which the authors explore the effect

[1] If the distribution is not known, and 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 asymptotic throughput, while analyzing adversarial strategies yields upper bounds.

Table 1 Summary of results presented

Arrivals	Feedback Deferred	Upper Bound 0	Lower Bound 0
Adversarial	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	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

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

of adversarial jamming on broadcasting in multiple-access channels under dynamic packet arrivals. They constrain both the arrival and jamming processes and give upper bounds on worst-case latency of widely used protocols. Last but not least, the work of Li (2011), tries to maximize the weighted throughput over a fading wireless channel considering packets with deadlines. They look at both the offline and online version of the problem and consider preemptive and non-preemptive scheduling. One difference with our work is that they consider uniform packet lengths with different weights, and their transmission time depends on the channel's quality (which changes with time). Moreover, instead of considering the transmission time for the metric, as we do, they consider the packets' weights.

We can also relate our work with the online version of the *bin packing* problem (Van Stee 2002), where the objects to be assigned to bins are the packets arriving to the sending station and the bins are the time intervals between two consecutive link failures. Some of the wide research that has taken place over the years around this problem, we consider to be related to ours. For example, Epstein and van Stee (2007) as well as Van Stee (2002) considered online bin packing with resource augmentation in the size of the bins, and used the so called *asymptotic performance ratio* for the evaluation of the competitiveness of the algorithm they propose. This metric corresponds to our *asymptotic throughput*, since they both follow the idea of long-term competitiveness. Observe that the essential difference of the online bin packing problem with the one that we are considering, is that in our system the bins and their sizes (intervals between failures) are unknown.

Another problem related to our work is the one of *buffer management*, see for example the survey of Goldwasser (2010), and the works of Li and Zhang (2009), Kogan et al. (2012) and Kogan et al. (2013). The theoretical community began applying the competitive analysis in this domain of work in 2000, with the works of Aiello et al. (2000), Mansour et al. (2000) and Kesselman et al. (2004). Focusing on the work of Li and Zhang (2009) in particular, it concentrates on a variant of the FIFO buffering model. Packets arrive dynamically and they are either sent or dropped due to the limited capacity of the buffer, say B. This can be seen as the corresponding jamming in our mode, but with a constant rate B.

Finally, the work of Jurdzinski et al. (2014) was motivated by the conference version of the present paper, and proposed an algorithm for scheduling packets of an arbitrary number of lengths, say k.

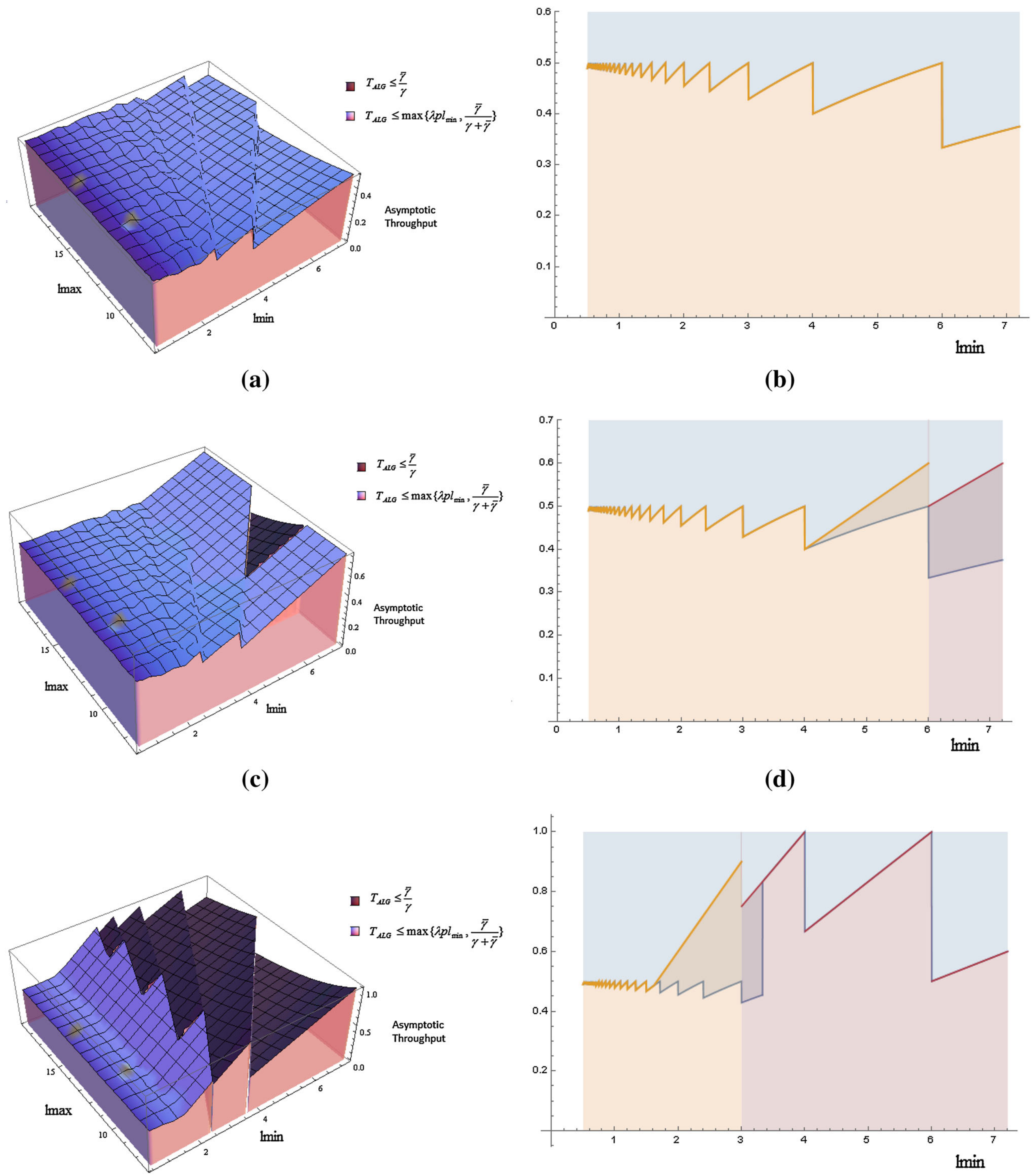
2 Model

2.1 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 of them associated with its arrival time (based on the station's local clock) and its length. We denote $\ell_{\min}$ and $\ell_{\max}$ to be the smallest and largest lengths, respectively, that a packet may have. We also use the notation $\gamma = \ell_{\max}/\ell_{\min}$, $\overline{\gamma} = \lfloor\gamma\rfloor$ and $\hat{\gamma} = \lceil\gamma\rceil - 1$. We assume that all packets are transmitted at the same bit rate through the link, hence the transmission time is proportional to the packet's length. However, the link is unreliable, that is, transmitted packets might be corrupted by bit errors.

2.2 Arrival models

– *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, $\ell_{\min}$ and $\ell_{\max}$, each packet that arrives is assigned one of the two lengths independently, with probabilities $p > 0$ and $q > 0$, respectively, where $p + q = 1$.



(a)

(b)

(c)

(d)

(e)

(f)

Fig. 2 In the *left column*, we give 3D representations of the upper bounds on the asymptotic throughput under stochastic packet arrivals for a range of $\ell_{\min}$ and $\ell_{\max}$ values. In the *right column*, we give 2D representations of the same graph, with additional information on the lower bound on the asymptotic throughput, under arbitrarily fixed $\ell_{\max}$. In *both columns*, we assume $\ell_{\min}$-packet arrival probabilities as follows: **a**, **b** $p = 0.01$, **c**, **d** $p = 0.1$ and **e**, **f** $p = 0.3$

2.3 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 considered to be unsuccessful, therefore the packet needs to be retransmitted in full. Even though we focus mainly on the *instantaneous feedback* mechanism for the notification of the sender about the error (the sending station is notified about the bit error as soon as it happens), in the case of *deferred feedback* the sending station is only notified about the error when the packet is fully received by the receiving end of the link.

2.4 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 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.

2.5 Efficiency metric

2.5.1 Asymptotic 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 transmitted (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 $E = E(A, t)$ that defines errors up to 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. We require that any pair of arrival and error patterns $A \in \mathcal{A}$ and $E \in \mathcal{E}$ must allow non-trivial communication, that is, the total length of transmitted packets is unbounded with t going to infinity; $\lim_{t\to\infty} L_X(A, E, t) = \infty$, for any algorithm X.

For arrival pattern A, adversarial error pattern E and time t, we define the *asymptotic throughput* $T_{\text{Alg}}(A, E, t)$ *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 $L_{\text{Alg}}(A, E, t) = L_{\text{OPT}}(A, E, t) = 0$. Then, we define the *asymptotic throughput* of algorithm Alg in the adversarial arrival model as:

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

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:

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

Note that the asymptotic throughput is different from the classical competitiveness ratio. In the classical competitive analysis, an algorithm Alg would be x-competitive if $L_{\text{Alg}}(A, E, t) \le x \times L_{\text{OPT}}(A, E, t) + \Delta$, for any t, OPT and patterns A and E, from which Δ is independent. The difference with the efficiency measure we described above, basically lies in the additive term Δ of the competitiveness formula which in our case may depend on time, and the fact that the final ratio is taken as the limit of the instantaneous ratio as time goes to infinity.

To prove lower bounds on the asymptotic throughput, we compare the performance of a given algorithm with that of

OPT. When deriving upper bounds, we compare the performance of some carefully chosen offline algorithm OFF with the performance of *any* online algorithm Alg. As we demonstrate later, this approach leads to accurate upper bound results.

Finally, we consider *work conserving* online scheduling algorithms, in the sense that, as long as there are pending packets, the sender does not cease to schedule them. 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 NP-hardness

We now prove the NP-hardness of the offline version of the scheduling problem we are studying in this work, defined for a single link as follows:

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 1 *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. □

4 Adversarial arrivals

This section focuses on adversarial packet arrivals. We first study the asymptotic throughput of any algorithm under the deferred feedback mechanism, to show the necessity of immediate feedback. Recall that with the deferred feedback the sending station is notified about a corrupted packet only after its full transmission.

Theorem 2 *No packet scheduling algorithm Alg can achieve an asymptotic throughput larger than* 0 *under adversarial arrivals in the deferred feedback model, even with one packet length.*

Proof Consider the case at which packets arrive frequently enough so that there are always some packets ready at the sender station, when it is about to make a decision. The algorithm will then greedily send a train of packets, while the adversary injects bit errors at a distance of exactly ℓ so that each error hits a different packet. Hence, the Alg 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, which results to an asymptotic throughput equal to 0 as claimed. □

We therefore focus on immediate feedback, for the rest of the section. 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 asymptotic 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. □

4.1 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 $\ell_{\max}$ and initially assume that there are no packets of length $\ell_{\min}$. 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 two 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 $\ell_{\max}$ packet (the first phase of the execution belongs to this class). Immediately after Alg starts transmitting the $\ell_{\max}$ packet, a set of $\hat{\gamma}$ $\ell_{\min}$-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 $\ell_{\max}$ 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 $\ell_{\min}$ packet. In this case, OFF transmits an $\ell_{\max}$ packet. Immediately after this transmission is completed, a link error occurs. Observe that in this phase Alg has transmitted successfully several $\ell_{\min}$ 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 $\ell_{\min}$ packets, for $j \in [1, \overline{\gamma}]$. Then, the asymptotic 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 $\ell_{\min}$ 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 asymptotic throughput increases with the average number of $\ell_{\min}$ packets transmitted in the phases of type 2. Hence, the throughput would be maximal if all the $\ell_{\min}$ packets are used in phases of type 2 with $\overline{\gamma}$ packets. With the above we obtain the following theorem.

Theorem 3 *The asymptotic 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 $\sum_{j=1}^{\overline{\gamma}} p_2(j) \geq \sum_{j=1}^{\overline{\gamma}} \frac{j p_2(j)}{\overline{\gamma}}$ in Eq (1), we get

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

which is a function that increases with $\sum_{j=1}^{\overline{\gamma}} j p_2(j)$. Since $\sum_{j=1}^{\overline{\gamma}} j p_2(j) \leq \hat{\gamma} p_1$, the asymptotic throughput can be bounded by

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

□

Two natural scheduling policies one could consider for this problem are the SL and LL algorithms; the first gives priority to $\ell_{\min}$ packets, whereas the second gives priority to the $\ell_{\max}$ packets. However, these two policies are not efficient in the considered setting. We prove that algorithm SL cannot have asymptotic throughput larger than $\frac{1}{\gamma+1}$ under adversarial arrivals. Algorithm LL is even worse; its asymptotic throughput cannot be more than 0 even under stochastic arrivals.

Theorem 4 *Algorithm* SL *cannot achieve asymptotic 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 $\ell_{\max}$ and one of length $\ell_{\min}$. SL schedules first the packet of length $\ell_{\min}$, and when it is transmitted, it schedules the packet of length $\ell_{\max}$. Meanwhile, an offline algorithm OFF schedules first the packet of length $\ell_{\max}$. When it is transmitted, the adversary causes an error on the link, so SL does not transmit successfully the packet of length $\ell_{\max}$. Now, SL only has one packet of length $\ell_{\max}$ in its queue (when this scenario is repeated will have several, but no packets of length $\ell_{\min}$). Hence, SL schedules this packet, while OFF schedules the packet of length $\ell_{\min}$ that has in its queue. When OFF completes the transmission of the $\ell_{\min}$ packet, the adversary causes an error on the link. This scenario can be repeated forever. In each instance, OFF transmits one packet of length $\ell_{\max}$ and one of length $\ell_{\min}$, while SL only transmits one packet of length $\ell_{\min}$. Hence, the throughput achieved in this execution 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. □

Theorem 5 *Algorithm* LL *cannot achieve asymptotic throughput larger than* 0*, even under stochastic arrivals.*

Proof The scenario is as follows. The adversary blocks all successful transmissions (by placing errors at distance smaller than $\ell_{\min}$) until at least two packets have arrived, one of length $\ell_{\max}$ and one of length $\ell_{\min}$. Algorithm LL schedules a packet of length $\ell_{\max}$, while an offline algorithm OFF schedules a packet of length $\ell_{\min}$. 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 $\ell_{\max}$ packet. Then, again the adversary blocks successful transmissions until OFF has at least one $\ell_{\min}$ packet pending. The scenario is repeated forever; while OFF will be transmitting successfully all $\ell_{\min}$ packets, LL will be stuck on the unsuccessful transmissions of $\ell_{\max}$ packets. Hence, the throughput will be 0. □

4.2 Lower bound and Algorithm SL-Preamble

We therefore propose algorithm SL-Preamble, that tries to combine in a graceful and efficient manner these two policies, SL and LL. It is a bit surprising, that their combination provides an optimal asymptotic throughput, while none of them is sufficiently good when considered on its own.

4.2.1 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 $\ell_{\min}$ 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 $\ell_{\min}$ packets available, it simply schedules packets following the LL policy.

4.2.2 Algorithm analysis

We show that algorithm SL-Preamble achieves an asymptotic 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 during which the link experiences no errors and the sender's queue of pending packets (in SL-Preamble) does not become empty. An inactive period is a non-active one. In other words, a time interval $T = [t_i, t_{i+1})$ is an active period if it starts with time instant t_i such that (a) it is the time of some task injection after an interval where the queue of SL-Preamble has been empty, or (b) it is the time right after an error in the link. Active period T ends with time instant t_{i+1} such that (i) it is the time at which an error occurs in the link, or (ii) it is the time at which the queue of pending packets becomes empty for SL-Preamble.

Note that in case (a) the corresponding inactive period had started when the queue of SL-Preamble became empty before time t_i, say at time instant t', and hence covers interval $[t', t_i)$. On the other hand, in case (b) and (i) hold, the corresponding inactive period will only be the time instant right before t_i, and hence neither SL-Preamble nor OPT can make any progress in transmitting a packet. Finally, in case (b) and (ii) hold, the corresponding inactive period will start at time t_{i+1} until new packets arrive at the sender, say at time instant t''. Observe that during the inactive periods it must be the case that the pending queue of OPT is also empty, otherwise it would contradict the optimality of OPT (recall that we consider offline algorithms being work conserving. OPT is also an offline algorithm, since it knows both arrival and error patterns from the beginning). 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 $\ell_{\min}$ packet and has length $L < \overline{\gamma}\ell_{\min}$.
2. Phase starting with $\ell_{\min}$ packet and length $L \geq \overline{\gamma}\ell_{\min}$.
3. Phase starting with $\ell_{\max}$ packet and has length $L < \ell_{\max}$.
4. Phase starting with $\ell_{\max}$ 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 $\ell_{\min}$ packets not in the preambles, b_i the number of successfully transmitted $\ell_{\max}$ packets, and c_i the number of successfully transmitted $\ell_{\min}$ packets in preambles. For the execution of OPT and within the ith phase, let a_i^* be the total number of successfully transmitted $\ell_{\min}$ packets and b_i^* the total number of successfully transmitted $\ell_{\max}$ 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 $\ell_{\max}$ (i.e., $C_O^2 - C_A^2 < \ell_{\max}$). Therefore, $C_A^2 \geq \frac{\ell_{\min}\overline{\gamma}}{\ell_{\max}+\ell_{\min}\overline{\gamma}} C_O^2$ (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

$$R_1 = \frac{\sum_{j \le i^*} C_A(j) + \sum_{j > i^*} C_A(j)}{\sum_{j \le i^*} C_O(j) + \sum_{j > i^*} C_O(j)}. \qquad (2)$$

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

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

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

$$\begin{aligned} T &= \lim_{j\to\infty} \frac{A + \frac{\ell_{\min}\overline{\gamma}}{\ell_{\max}+\ell_{\min}\overline{\gamma}} \sum_{j > i^*} C_O(j)}{O + \sum_{j > i^*} C_O(j)} \\ &= \lim_{j\to\infty} \frac{(\ell_{\max} + \ell_{\min}\overline{\gamma})A + (\ell_{\min}\overline{\gamma}) \sum_{j > i^*} C_O(j)}{(\ell_{\max} + \ell_{\min}\overline{\gamma})(O + \sum_{j > i^*} C_O(j))} \\ &= \lim_{j\to\infty} \Bigg(\frac{\ell_{\min}\overline{\gamma}}{\ell_{\max} + \ell_{\min}\overline{\gamma}} \\ &\quad + \frac{(\ell_{\max} + \ell_{\min}\overline{\gamma})A - (\ell_{\min}\overline{\gamma})O}{(\ell_{\max} + \ell_{\min}\overline{\gamma})(O + \sum_{j > i^*} C_O(j))} \Bigg) \\ &= \frac{\ell_{\min}\overline{\gamma}}{\ell_{\max} + \ell_{\min}\overline{\gamma}} \\ &= \frac{\overline{\gamma}}{\gamma + \overline{\gamma}}. \end{aligned}$$

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

So far, we have basically seen what is the asymptotic 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 $C_A \ge \frac{\ell_{\min}\overline{\gamma}}{\ell_{\max}+\ell_{\min}\overline{\gamma}} C_O$, we know that $X_t \ge \frac{\ell_{\min}\overline{\gamma}}{\ell_{\max}+\ell_{\min}\overline{\gamma}} Y_t$ as well. Therefore,

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

This completes the lower bound of asymptotic 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 $\ell_{\min}$ and $\ell_{\max}$ 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 $\ell_{\min}$ packets successfully sent by time t by OPT is no more than the amount of $\ell_{\min}$ packets transmitted by SL-Preamble plus $\overline{\gamma} - 1$, i.e., $\sum_{i<j} a_i^* \le \sum_{i<j}(a_i + c_i) + (\overline{\gamma} - 1)$.*

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

Lemma 2 *Considering all kinds of phases and the number of $\ell_{\max}$ packets, $\sum_{i \le j} b_i^* \le \sum_{i \le j} b_i + \sum_{i \le j} \frac{c_i}{\overline{\gamma}} + 2$, for every 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

$$\sum_{i \le j-1} b_i^* \le \sum_{i \le j-1} b_i + \sum_{i \le j-1} \frac{c_i}{\overline{\gamma}} + 2\,.$$

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 $\ell_{\max}$ packets pending. Let t be the latest such time in the phase. Let us define $b^*(t)$ and $b(t)$ being the number of $\ell_{\max}$ packets successfully transmitted up to time t by OPT and SL-Preamble, respectively. We know that $b^*(t) \le b(t)$. Let also $x_j^*(t)$ and $x_j(t)$ be the number of $\ell_{\max}$ packets sent by OPT and SL-Preamble, respectively, after time point t until the end of the phase j. We claim that $x_j^*(t) \le x_j(t) + 2$. From our definitions, at time t SL-Preamble is transmitting a $\ell_{\min}$ packet. Since t is the last time that SL-Preamble has no $\ell_{\max}$ packet in its queue, the worst case is being at the beginning of the preamble (by inspection of the four types of phases). Then, if the phase ends at time t', we define period $I = [t, t']$ such that:

$$|I| < \overline{\gamma}\ell_{\min} + (x_j(t) + 1)\ell_{\max} \le (x_j(t) + 2)\ell_{\max}.$$

The +1 $\ell_{\max}$ packet is because of the link failure before transmitting completely the last $\ell_{\max}$ scheduled packet of the phase. Observe that OPT could be transmitting a $\ell_{\max}$ packet at time t, received by the receiver at some point in $[t, t+\ell_{\max}]$ and accounted for in $x_j^*(t)$. Therefore,

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

Now consider the case where at all times of a phase j there are $\ell_{\max}$ packets in the queue of SL-Preamble. By inspection of the four types of phases, the worst case is when j is of type 2. Since there is always some $\ell_{\max}$ packet pending in SL-Preamble, after sending the $\overline{\gamma}\ell_{\min}$ packets, it will keep scheduling $\ell_{\max}$ packets, until a link failure corrupts the last one scheduled, or the queue becomes empty. On the same time OPT is able to successfully transmit at most $\lfloor \frac{L_j}{\ell_{\max}} \rfloor \le b_j + 1$ packets of length $\ell_{\max}$, where L_j is the length of the phase. Therefore, in all types of phases, $b_j^* \le \frac{c_j}{\overline{\gamma}} + b_j$. And hence by induction the claim follows; $\sum_{i \le j} b_i^* \le \sum_{i \le j} \frac{c_i}{\overline{\gamma}} + \sum_{i \le j} b_i + 2$. □

Combining the two lemmas above, Lemmas 1 and 2:

$$\begin{aligned}
R_2 &= \frac{\sum_{i \le j} C_A(i)}{\sum_{i \le j} C_O(j)} = \frac{\sum_{i \le j} [(a_i + c_i)\ell_{\min} + b_i \ell_{\max}]}{\sum_{i \le j} \left[a_i^* \ell_{\min} + b_i^* \ell_{\max}\right]} \\
&\ge \frac{\sum_{i \le j} [(a_i + c_i)\ell_{\min} + b_i \ell_{\max}]}{\sum_{i \le j} (a_i + c_i)\ell_{\min} + (\overline{\gamma} - 1)\ell_{\min} + \sum_{i \le j} (b_i + \frac{c_i}{\overline{\gamma}})\ell_{\max} + 2\ell_{\max}} \\
&\ge \frac{\sum_{i \le j} [(a_i + c_i)\ell_{\min} + b_i \ell_{\max}]}{\sum_{i \le j} [(a_i + 2c_i)\ell_{\min} + b_i \ell_{\max}] + 3\ell_{\max}} \\
&\ge \frac{\sum_{i \le j} [(a_i + c_i)\ell_{\min} + b_i \ell_{\max}] + \frac{3}{2}\ell_{\max} - \frac{3}{2}\ell_{\max}}{2\sum_{i \le j} [(a_i + c_i)\ell_{\min} + b_i \ell_{\max}] + 3\ell_{\max}} \\
&\ge \frac{1}{2} - \frac{\frac{3}{2}\ell_{\max}}{2\sum_{i \le j} [(a_i + c_i)\ell_{\min} + b_i \ell_{\max}] + 3\ell_{\max}}.
\end{aligned}$$

Note that, due to parameters a_i, b_i and c_i the second ratio tends to zero (the denominator tends to infinity while the nominator is constant). Therefore,

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

Theorem 6 *The asymptotic 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 $\frac{\overline{\gamma}}{\gamma + \overline{\gamma}} \le \frac{1}{2}$ it is easy to conclude that the asymptotic throughput of Algorithm SL-Preamble is at least $\frac{\overline{\gamma}}{\gamma + \overline{\gamma}}$ as claimed. □

5 Stochastic arrivals

We now turn our attention to stochastic packet arrivals. We first consider the deferred feedback mechanism and show that also in this case the upper bound on the asymptotic throughput is 0.

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

Proof As described in Sect. 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. □

The rest of the section is therefore focused on analyzing of the immediate feedback mechanism.

5.1 Upper bounds

In order to find the upper bound of the asymptotic 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 $\ell_{\min}$ and $\ell_{\max}$ (recall the definitions of λ, p and q from Sect. 2). We define the following adversarial error model $\mathcal{E}$.

1. When Alg starts a phase by transmitting an $\ell_{\max}$ packet then,
   (a) If OFF has $\ell_{\min}$ packets pending, then the adversary extends the phase so that OFF can transmit successfully as many $\ell_{\min}$ packets as possible, up to $\hat{\gamma}$. Then, it ends the phase so that Alg does not complete the transmission of the $\ell_{\max}$ packet (since $\hat{\gamma}\ell_{\min} < \ell_{\max}$).
   (b) If OFF does not have any $\ell_{\min}$ 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 $\ell_{\min}$ packet then,
   (a) IF OFF has a packet of length $\ell_{\max}$ pending, then the adversary extends the phase so OFF can transmit an $\ell_{\max}$ 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 $\ell_{\min}$.
   (b) If OFF has no $\ell_{\max}$ 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 $\ell_{\min}$ are sent by OFF completed by time t. The value $r_{2a}^{(j)}$ with $j \leq \overline{\gamma}$ packets of length $\ell_{\min}$ 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 asymptotic 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 8 *No algorithm Alg has asymptotic throughput larger than* $\frac{\overline{\gamma}}{\gamma}$.

Proof It can be observed in Eq. 4 that, for a fixed r, the lower the value of $r_{1a}^{(j)}$ the higher the asymptotic 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* $\ell_{\min}$ *(resp.,* $\ell_{\max}$*) 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* $\ell_{\min}$ *(resp.,* $\ell_{\max}$*) 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 $\ell_{\min}$ has parameter λp. By the definition of a Poisson process, the distribution of packets of length $\ell_{\min}$ 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), cf., (Mitzenmacher and Upfal 2005), 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 $\ell_{\max}$ 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, cf., (Mitzenmacher and Upfal 2005). □

Now we can show the following result.

Theorem 9 *Let* $p < q$. *Then, the asymptotic 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 8. 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 such constants always exist. Then, we prove that the asymptotic 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 $\ell_{\min}$ and $\ell_{\max}$ packets, respectively, injected up to time t. Let r_t^{off} and s_t^{off} be the number of $\ell_{\max}$ and $\ell_{\min}$ packets, respectively, successfully transmitted by OFF by time t. Similarly, let s_t^{alg} be the number of $\ell_{\min}$

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 $\ell_{\min}$ 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 $\ell_{\min}$ 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 not smaller than $\ell_{\min}$ at which OFF has no $\ell_{\min}$ 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 $\ell_{\min}$ packets. The asymptotic throughput at time t_i can be bounded as

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

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

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

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 $\ell_{\min}$ packets. Recall that for any $t \geq \ell_{\min}$, from Lemma 3, we have that the number of $\ell_{\min}$ packets injected by time t satisfy $a_t^{\min} > \eta' pt$ with probability at most $e^{-c't}$ and the injected max packets satisfy $a_t^{\max} < \eta qt$ with probability at most e^{-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 $1 - e^{-c't} - e^{-ct}$. If this holds, it implies that OFF will always have $\ell_{\max}$ 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 $\eta' p i \delta$ packets $\ell_{\min}$ are injected is no more than $e^{-c'' i \delta}$.

With the above, the asymptotic 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 $1 - e^{-ct_i} - e^{-c't_i} - e^{-c''t_i}$. Observe that as i goes to infinity the above bound converges to $\eta' p \ell_{\min}$, while the probability converges exponentially fast to 1. □

We now prove that algorithm SL cannot have asymptotic throughput larger than $\frac{1}{\gamma+1}$ under stochastic arrivals with specific arrival rates. This motivates the need for devising a new online algorithm for packet scheduling in these cases.

Theorem 10 *$\forall \varepsilon > 0, \exists \lambda, p, q$ such that algorithm SL cannot achieve an asymptotic 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 $\ell_{\min}$ packet. Then, I_i starts as soon as an $\ell_{\min}$ 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 $\ell_{\max}$ packet in its queue at the start of the interval or not. If it has an $\ell_{\max}$ 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 $\ell_{\min}$ packet. In a short interval, OFF sends an $\ell_{\min}$ packet, followed by an error injected by the adversary. Hence, in a short interval both SL and OFF successfully transmit one $\ell_{\min}$ packet. In a long interval, OFF sends an $\ell_{\max}$ packet, after which the adversary injects an error. (Up to that point SL has been able to complete the transmission of one or more $\ell_{\min}$ packets, but no $\ell_{\max}$ packet.) After the error, OFF sends an $\ell_{\min}$ 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 $\ell_{\min}$ packet and one $\ell_{\max}$ packet, while SL transmits only $\ell_{\min}$ 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 $\ell_{\min}$ packets that arrive up to the end of interval I_k is bounded by k (that accounts for the $\ell_{\min}$ 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 $\ell_{\min}$ 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 $\ell_{\min}$ 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 $1 - e^{-\delta^2(1-e^{-\lambda q \ell_{\min}})x/2}$, for any $\delta \in (0,1)$.

Proof of Claim Observe that if an $\ell_{\max}$ 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 $\ell_{\min}$, some $\ell_{\max}$ 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 $\ell_{\max}$ packets arrive is at most $e^{-\delta^2(1-e^{-\lambda q \ell_{\min}})x/2}$. This completes the proof of the Claim. □

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

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

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

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

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 $(1-\delta)(1-e^{-\lambda q \ell_{\min}}) \geq (1-\varepsilon)$).

5.2 Lower bound and Algorithm CSL-Preamble

In this section we propose algorithm CSL-Preamble (stands for Conditional SL-Preamble), which builds on algorithm SL-Preamble presented in Sect. 4.2, 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 $\ell_{\min}$ 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 11 *The asymptotic 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 6, asymptotic throughput of at least $\frac{\overline{\gamma}}{\gamma+\overline{\gamma}}$ under *any* error pattern.

Case $\frac{\overline{\gamma}}{2\gamma} < \lambda p \ell_{\min} \leq 1$. Our goal is to prove that the asymptotic throughput is not smaller than $\min\left\{\eta p \ell_{\min}, \frac{\overline{\gamma}}{\gamma}\right\}$, for any $\eta = \delta\lambda$, with $\delta < 1$. Considering such an η, we can make use of Lemma 3 with respect to λ, η, p. The asymptotic 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. Observe that any time instant t in the modified time line, say $t = t_m$, cannot be larger than the corresponding time t in the global time line, say t_g (i.e., $t_m \leq t_g$). 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 $t_i = i \times \ell_{\max}$. 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 $t_m \leq t_g$, 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 $1 - e^{-ct_i}$.

Consider an integer $j > 1$ being a square of another integer. We prove that by time t_j, the asymptotic throughput is at least

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

with probability at least $1-c'e^{-ct\sqrt{j}}$, 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 $\ell_{\min}$, 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 $\ell_{\min}$ at time t_j. With probability at least $1-c'e^{-ct\sqrt{j}}$, for every $\sqrt{j} \le i \le j$ 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 $\sum_{i\ge\sqrt{j}} e^{-ct_i} \le c' \times e^{-ct\sqrt{j}}$ 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 $\bigcup_{i=\sqrt{j}}^{j} S_i$; executions at which all S_i events happen, for $\sqrt{j} \le i \le j$. Using induction on i, we prove the following claim:

Claim At least $t_i\eta p-\overline{\gamma}$ packets of length $\ell_{\min}$ have been successfully transmitted by time t_i, or at least $\overline{\gamma}$ packets of length $\ell_{\min}$ are successfully transmitted in the interval $[t_i, t_{i+1}]$.

Proof of Claim First, recall that algorithm CSL-Preamble runs the SL policy, since $\lambda p\ell_{\min} > \frac{\overline{\gamma}}{2\gamma}$. Hence, as long as there are $\ell_{\min}$ packets pending, it will schedule them for transmission. Recall also, that times t_i represent time instants such that $t_i = i \times \ell_{\max}$ in the modified time line.

Base case: By time $t_{\sqrt{j}}$ and with probability at least $1-e^{ct\sqrt{j}}$, there will be at least $\eta pt_{\sqrt{j}}$ packets of length $\ell_{\min}$ arriving. Now since $t_{\sqrt{j}+1} = t_{\sqrt{j}} + \ell_{\max}$, if there are at least $\overline{\gamma}$ pending packets of length $\ell_{\min}$ at time $t_{\sqrt{j}}$, they will be successfully transmitted during the interval $[t_{\sqrt{j}}, t_{\sqrt{j}+1}]$, which guarantees the invariant. Otherwise, there are at least $\eta pt_{\sqrt{j}}-\overline{\gamma}$ pending packets of length $\ell_{\min}$ at time $t_{\sqrt{j}}$ (as many as the ones that arrived minus the completed ones since the beginning of the execution, in a duration of $\sqrt{j} \times \ell_{\max}$ time).

Induction hypothesis: For $\sqrt{j} < k < j$, the invariant holds.

Induction step: We will show that the invariant holds for $k+1$. Since we consider only executions in the union $\bigcup_{i=\sqrt{j}}^{j} S_i$, we know that by time $t_{k+1} = (k+1) \times \ell_{\max}$, there are at least ηpt_{k+1} packets of length $\ell_{\min}$ arriving, with probability at least $1-c'e^{-ct\sqrt{j}}$. Now, since $t_{k+2} = t_{k+1} + \ell_{\max}$, if there are at least $\overline{\gamma}$ pending packets of length $\ell_{\min}$ at time t_{k+1}, they will be successfully transmitted during the interval $[t_{k+1}, t_{k+2}]$. Otherwise, there are at least $\eta pt_{k+1} - \overline{\gamma}$ pending packets of length $\ell_{\min}$ at time t_{k+1} (as many as the ones that arrived minus the completed ones since the beginning of the execution, in a duration of $(k+1) \times \ell_{\max}$ time). This guarantees the invariant and hence completes the proof of the Claim. □

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 $\ell_{\min}$ pending in time t_i; if such an i does not exist, we set $i^* = -1$. Consider two sub-cases:

Sub-case $i^* = -1$ (i^* *does not exist*). Note that, by definition of i^*, at every time $t_i \in [\sqrt{j}, j]$, there are at least $\overline{\gamma}$ pending packets of length $\ell_{\min}$ pending. Consequently, by the specification of the algorithm CSL-Preamble, in each interval $[t_i, t_{i+1}]$, for $\sqrt{j} \le i < j$, at least $\overline{\gamma}$ packets of length $\ell_{\min}$ finish their transmission successfully. Therefore, by time t_j the total length of $\ell_{\min}$-packets successfully transmitted by algorithm CSL-Preamble is at least

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

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

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

which converges to $\frac{\overline{\gamma}}{\gamma}$ with j going to infinity.

Sub-case $i^* \in [\sqrt{j}, j]$. It 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 $\ell_{\min}$, and in each interval $[t_i, t_{i+1}]$, for $i^* \le i < j$, at least $\overline{\gamma}$ packets of length $\ell_{\min}$ finish their transmission successfully. Therefore, by time t_j the total length of $\ell_{\min}$-packets successfully transmitted by algorithm CSL-Preamble is at least

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

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

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

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

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

which converges to $\min\left\{\eta p \ell_{\min}, \frac{\underline{\gamma}}{\gamma}\right\}$ with j going to infinity. For the proof of inequality (*) in the above expression see the Lemmas 4 and 5 in the Appendix.

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 asymptotic throughput, since the subsequent ratios hold with probabilities approaching 1 exponentially fast (in j), i.e., with probabilities at least $1 - c' e^{-ct\sqrt{j}}$, where c and c' are positive constants dependent only on λ, η, p. The minimum of the two asymptotic throughputs, coming from the sub-cases, is $\min\left\{\eta p \ell_{\min}, \frac{\underline{\gamma}}{\gamma}\right\}$, as desired and therefore the asymptotic throughput is at least $\min\left\{\delta\lambda p \ell_{\min}, \frac{\underline{\gamma}}{\gamma}\right\}$ in this case.

Case $\lambda p \ell_{\min} > 1$. In this case, we simply observe that we get at least the same asymptotic throughput as in case $\lambda p \ell_{\min} = 1$, because we are dealing with executions saturated with packets of length $\ell_{\min}$ 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 asymptotic throughput in this case is at least $\min\left\{\eta p \ell_{\min}, \frac{\underline{\gamma}}{\gamma}\right\}$, for any $\lambda/2 < \eta < \lambda$, and therefore it is at least $\min\left\{\lambda p \ell_{\min}, \frac{\underline{\gamma}}{\gamma}\right\} \geq \min\left\{1, \frac{\underline{\gamma}}{\gamma}\right\} = \frac{\underline{\gamma}}{\gamma}$.

Observe that if we compare the upper bounds on asymptotic 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 asymptotic throughput). In the case where γ is not an integer, there is a small gap between the upper and lower bound results.

6 Randomized algorithms

So far we have considered deterministic solutions. In many problems considered in computer science, randomized solutions can obtain better performance. Using Yao's principle, Yao (1977), we show that this is not the case for the problem considered in this work.

For arrival pattern A, adversarial error-function E, string of random bits R and time t, we define the *asymptotic throughput* $T_{\text{Alg}}(A, E, R, t)$ *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 $L_{\text{Alg}}(A, E, R, t) = L_{\text{OPT}}(A, E, R, t) = 0$. And we define the *asymptotic throughput* of algorithm Alg in the adversarial arrival model as follows:

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

where $\mathcal{R}$ is a distribution of all possible strings of random bits used by the algorithm. In the stochastic arrival model, the asymptotic throughput needs to take into account the random distribution of arrival patterns in $\mathcal{A}$ and it is defined as follows:

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

Based on the above definitions, we apply now Yao's principle, Yao (1977), to obtain the following result.

Observation 1 *All upper bounds found for deterministic algorithms in* Sects. 4 and 5 *with instantaneous feedback, hold also for randomized algorithms, even for oblivious adversaries.*

Yao's principle states the following: Given an online problem, let $c_{\mathcal{R}}$ be the smallest competitive ratio of randomized online algorithm $\mathcal{R}$ against any oblivious adversary. Let also P be a probability distribution for the input sequence, such that c_A^P is the smallest competitive ratio of deterministic online algorithm A under P. Then, the competitive ratio of the best randomized algorithm against any oblivious adversary, is equal to the competitive ratio of the best deterministic online algorithm under a worst-case input distribution, i.e., $\inf_{\mathcal{R}} c_{\mathcal{R}} = \sup_P \inf_A c_A^P$.

7 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 asymptotic throughput in the 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 considered model, for example, considering more elaborated/realistic distributions for the packet arrival, assuming that packets that suffer errors are not retransmitted [which applies when forward error correction (Raghavan et al. 2001) 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 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 asymptotic throughput. Finally, we believe that the definition of asymptotic throughput as proposed here can be adapted, possibly in a different context, to other metrics and problems.

Acknowledgments The authors would like to thank the anonymous reviewers that their constructive feedback has helped to significantly improve the manuscript. This research was partially supported by the Madrid Regional Government (CM) through the TIGRE5-CM program (S2013/ICE-2919) and the Grant Cloud4BigData (S2013/ICE-2894, cofunded by FSE and FEDER), by the Spanish Ministry of Economy and Competitiveness with Grant TEC2014-55713-R and the Ramon y Cajal Grant (RYC-2012-10788), and by the Spanish Ministry of Education, Culture and Sports (MECD) with the Grant FPU12/00505.

Appendix

Lemma 4 *When* $\eta p \ell_{\min} \leq \frac{\overline{\gamma}}{\gamma}$ *it holds that* $\frac{(t_{i^*}\eta p - \overline{\gamma})\ell_{\min} + \frac{t_j - t_{i^*}}{\ell_{\max}}\overline{\gamma}\ell_{\min}}{t_j} \geq \frac{(t_j \eta p - \overline{\gamma})\ell_{\min}}{t_j}$.

Proof Let us assume first the case of $i^* < j$. This means that:

$$\eta p \ell_{\min} \leq \frac{\overline{\gamma}}{\gamma} = \frac{(j - i^*)\overline{\gamma}}{(j - i^*)\gamma} = \frac{(j - i^*)\overline{\gamma}\ell_{\min}}{(j - i^*)\ell_{\max}}$$
$$\Rightarrow \eta p \ell_{\min}(i^* - j)\ell_{\max} + (j - i^*)\overline{\gamma}\ell_{\min} \geq 0$$
$$\Rightarrow i^*\ell_{\max}\eta p \ell_{\min} + (j - i^*)\overline{\gamma}\ell_{\min} \geq j\ell_{\max}\eta p \ell_{\min}$$
$$\Rightarrow t_{i^*}\eta p \ell_{\min} + \frac{t_j - t_{i^*}}{\ell_{\max}}\overline{\gamma}\ell_{\min} \geq t_j \eta p \ell_{\min}$$
$$\Rightarrow (t_{i^*}\eta p - \overline{\gamma})\ell_{\min} + \frac{t_j - t_{i^*}}{\ell_{\max}}\overline{\gamma}\ell_{\min} \geq (t_j \eta p - \overline{\gamma})\ell_{\min}.$$

What is more, for the case when $i^* = j$, we have that:

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

Both cases conclude to the same, which proves the lemma. □

Lemma 5 *When* $\eta p \ell_{\min} > \frac{\overline{\gamma}}{\gamma}$ *it holds that* $\frac{(t_{i^*}\eta p - \overline{\gamma})\ell_{\min} + \frac{t_j - t_{i^*}}{\ell_{\max}}\overline{\gamma}\ell_{\min}}{t_j} \geq \frac{\frac{(t_j - t_{\sqrt{j}})}{\ell_{\max}}\overline{\gamma}\ell_{\min}}{t_j}$.

Proof When $\eta p \ell_{\min} > \frac{\overline{\gamma}}{\gamma}$, the following is also true:

$$\eta p \ell_{\min} \geq \frac{\overline{\gamma}}{\gamma} + \frac{(1 - \sqrt{j})\overline{\gamma}}{i^*\gamma}.$$

This means that:

$$\eta p \ell_{\min} \geq \frac{(1 + i^* - \sqrt{j})\overline{\gamma}\ell_{\min}}{i^*\ell_{\max}}$$
$$\Rightarrow \eta p \ell_{\min} i^* \ell_{\max} + \overline{\gamma}\ell_{\min}(j - i^* - 1 - j + \sqrt{j}) \geq 0$$
$$\Rightarrow i^*\ell_{\max}\eta p \ell_{\min} - \overline{\gamma}\ell_{\min} + (j - i^*)\overline{\gamma}\ell_{\min} \geq (j - \sqrt{j})\overline{\gamma}\ell_{\min}$$
$$\Rightarrow (t_{i^*}\eta p - \overline{\gamma})\ell_{\min} + \frac{t_j - t_{i^*}}{\ell_{\max}}\overline{\gamma}\ell_{\min} \geq \frac{(t_j - t_{\sqrt{j}})}{\ell_{\max}}\overline{\gamma}\ell_{\min}.$$

□

References

Aiello, W. A., Mansour, Y., Rajagopolan, S., & Rosén, A. (2000). Competitive queue policies for differentiated services. In *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE* (vol. 2, pp. 431–440). IEEE.

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

Anantharamu, L., Chlebus, B. S., Kowalski, D. R., & Rokicki, M. A. (2011). Medium access control for adversarial channels with jamming. In A. Kosowski & M. Yamashita (Eds.), *Structural Information and Communication Complexity* (pp. 89–100). Berlin: Springer.

Andrews, M., & Zhang, L. (2005). Scheduling over a time-varying user-dependent channel with applications to high-speed wireless data. *Journal of the ACM*, *52*(5), 809–834. doi:10.1145/1089023.1089028. ISSN 0004-5411.

Awerbuch, B., Kutten, S., & Peleg, D. (1992). Competitive distributed job scheduling. In *Proceedings of the twenty-fourth annual ACM symposium on Theory of computing* (pp. 571–580). ACM.

Becchi, M. (2008). *From poisson processes to self-similarity: a survey of network traffic models*. Technical report, Citeseer.

Borodin, A., & El-Yaniv, R. (1998). *Online computation and competitive analysis*. Cambridge: Cambridge University Press.

Epstein, L., & van Stee, R. (2007). Online bin packing with resource augmentation. *Discrete Optimization*, *4*(34), 322–333. doi:10.1016/j.disopt.2007.09.004. (ISSN 1572-5286).

Goldwasser, M. H. (2010). A survey of buffer management policies for packet switches. *ACM SIGACT News*, *41*(1), 100–128.

Jamieson, K., & Balakrishnan, H. (2007). PPR: Partial packet recovery for wireless networks. In *Proceedings of the 2007 conference on Applications, technologies, architectures, and protocols for computer communications*, SIGCOMM '07 (pp. 409–420), New York, NY, USA, ACM. ISBN 978-1-59593-713-1. doi:10.1145/1282380.1282426.

Jurdzinski, T., Kowalski, D. R., & Lorys, K. (2014). Online packet scheduling under adversarial jamming. *Approximation and online algorithms* (pp. 193–206). Berlin: Springer.

Kesselheim, T. (2012). Dynamic packet scheduling in wireless networks. In *PODC*, (pp. 281–290)

Kesselman, A., Lotker, Z., Mansour, Y., Patt-Shamir,B., Schieber, B., & Sviridenko, M. (2004). Buffer overflow management in qos switches. *SIAM Journal on Computing*, *33*(3), 563–583.

Kogan, K., Lpez-Ortiz, A., Nikolenko, S. I., Sirotkin, A. V., & Tugaryov, D. (2012). Fifo queueing policies for packets with heterogeneous processing. In G. Even & D. Rawitz (Eds.), *Design and analysis of algorithms* (Vol. 7659, pp. 248–260). Lecture Notes in Computer Science Berlin Heidelberg: Springer. doi:10.1007/978-3-642-34862-4. (ISBN 978-3-642-34861-7).

Kogan, K., Lopez-Ortiz, A., Nikolenko, S. I., & Sirotkin, A. V. (2013). Multi-queued network processors for packets with heterogeneous processing requirements. In *Communication Systems and Networks (COMSNETS), 2013 Fifth International Conference on*, pp. 1–10. doi:10.1109/COMSNETS.2013.6465538.

Li, F. (2011). A near-optimal memoryless online algorithm for fifo buffering two packet classes. In W. Wang, X. Zhu & D.-Z. Du (Eds.), *Combinatorial Optimization and Applications* (pp. 222–233). Berlin: Springer.

Li, F., & Zhang, Z. (2009). Scheduling weighted packets with deadlines over a fading channel. In *Information Sciences and Systems, 2009. CISS 2009. 43rd Annual Conference on*, (pp. 707–712). IEEE.

Lin, S., & Costello, D. J. (2004). *Error control coding* (Vol. 123). Englewood Cliffs, NJ: Prentice-hall.

Mansour, Y., Patt-Shamir, B., & Lapid, O. (2000). Optimal smoothing schedules for real-time streams. In *Proceedings of the nineteenth annual ACM symposium on Principles of distributed computing* (pp. 21–29). ACM.

Meiners, C., & Torng, E. (2007). *Mixed criteria packet scheduling*., Algorithmic aspects in information and management Berlin Heidelberg: Springer.

Mitzenmacher, M., & Upfal, E. (2005). *Probability and computing*. Cambridge: Cambridge University Press.

Pinedo, M. L. (2012). *Scheduling: Theory, algorithms, and systems*. Berlin: Springer.

Pruhs, K., Sgall, J., & Torng, E. (2004). Online scheduling. In J. Leung (Ed.), *Handbook of scheduling: Algorithms, Models and Performance analysis* (pp. 15-1–15-41). Boca Raton, FL: CRC Press.

Raghavan, A., Ramchandran, K., & Kozintsev, I. (2001). Continuous error detection (CED) for reliable communication. *IEEE Transactions on Communications*, *49*(9), 1540–1549.

Richa, A., Scheideler, C., Schmid, S., & Zhang, J. (2012). Competitive throughput in multi-hop wireless networks despite adaptive jamming. *Distributed Computing*, *26*, (159–171).

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

Van Stee, R. (2002). *On-line scheduling and bin packing*, IPA dissertation series. https://books.google.com.cy/books?id=1UzMGAAACAAJ.

Yao, A. C.-C. (1977). Probabilistic computations: Toward a unified measure of complexity. In *Proceedings of the 18th Annual Symposium on Foundations of Computer Science*, SFCS '77, (pp. 222–227), Washington, DC, USA. IEEE Computer Society. doi:10.1109/SFCS.1977.24.