On the Performance of Greedy Algorithms in Packet Buffering*

Susanne Albers† Markus Schmidt‡

Abstract

We study a basic buffer management problem that arises in network switches. Consider m input ports, each of which is equipped with a buffer (queue) of limited capacity. Data packets arrive online and can be stored in the buffers if space permits; otherwise packet loss occurs. In each time step the switch can transmit one packet from one of the buffers to the output port. The goal is to maximize the number of transmitted packets. Simple arguments show that any reasonable algorithm, which serves any non-empty buffer, is 2-competitive. Azar and Richter recently presented a randomized online algorithm and gave lower bounds for deterministic and randomized strategies.

In practice greedy algorithms are very important because they are fast, use little extra memory and reduce packet loss by always serving a longest queue. In this paper we first settle the competitive performance of the entire family of greedy strategies. We prove that greedy algorithms are not better than 2-competitive no matter how ties are broken. Our lower bound proof uses a new recursive construction for building adversarial buffer configurations that may be of independent interest. We also give improved lower bounds for deterministic and randomized online algorithms.

In the second part of the paper we present the first deterministic online algorithm that is better than 2-competitive. We develop a modified greedy algorithm, called *Semi-Greedy*, and prove that it achieves a competitive ratio of $17/9 \approx 1.89$. The new algorithm is simple, fast and uses little extra memory. Only when the risk of packet loss is low, it does not serve the longest queue. Additionally we study scenarios when an online algorithm is granted additional resources. We consider resource augmentation with respect to memory and speed, i.e. an online algorithm may be given larger buffers or higher transmission rates. We analyze greedy and other online strategies.

1 Introduction

The performance of high-speed networks critically depends on switches that route data packets arriving at the input ports to the appropriate output ports so that the packets can reach their correct destinations in the network. To reduce packet loss when the traffic is bursty, ports are equipped with buffers where packets can be stored temporarily. However the buffers are of limited capacity so that effective buffer management strategies are important to maximize the throughput at a switch. As a result there has recently been considerable research interest in the design and analysis of various buffer management policies [1–11].

We study a very basic problem in this context. Consider m input ports which serve a given output port. Each input port has a buffer that can simultaneously store up to B packets and is organized as a queue. In any time step new packets may arrive at the input ports and can be appended to the corresponding buffers if

*A preliminary version of this paper appeared at the *36th Annual ACM Symposium on Theory of Computing (STOC'04)*.

†Institut für Informatik, Albert-Ludwigs-Universität Freiburg, Georges-Köhler-Allee 79, 79110 Freiburg, Germany. salbers@informatik.uni-freiburg.de Work supported by the Deutsche Forschungsgemeinschaft, project AL 464/4-1, and by the EU, projects APPOL and APPOL II.

‡Institut für Informatik, Albert-Ludwigs-Universität Freiburg, Georges-Köhler-Allee 79, 79110 Freiburg, Germany. markus.schmidt@informatik.uni-freiburg.de

space permits. More specifically, suppose that the buffer at port i currently stores b_i packets and that a_i new packets arrive there. If $b_i + a_i \leq B$, then all new packets can be accepted; otherwise $a_i + b_i - B$ packets must be dropped. In any time step the switch can select one non-empty buffer and transmit the packet at the head through the output port. We assume w.l.o.g. that the packet arrival step precedes the transmission step. The goal is to maximize the *throughput*, i.e. the total number of transmitted packets. The scenario we study here arises, for instance, in input-queued (IQ) switches which represent the dominant switch architecture today. In an IQ switch with m input and m output ports packets that arrive at input i and have to be routed to output j are buffered in a virtual output queue Q_{ij}. In each time step, for any output j, one data packet from queues Q_{ij}, $1 \leq i \leq m$, can be sent to that output. The buffer size B is large, typically several hundreds or thousands. We emphasize that we consider all packets to be equally important, i.e. they all have the same value. Most current networks, in particular IP networks, treat packets from different data streams equally in intermediate switches.

Information on future packet arrivals usually is very limited or not available at all. We make no probabilistic assumptions about the input and investigate an online setting where at any time future packet arrivals are unknown. We are interested in online buffer management strategies that have a provably good performance. Following [12] we call a deterministic online algorithm *ALG* c-competitive if $c \cdot T_{ALG}(\sigma) \geq T_{OPT}(\sigma)$, for all packet arrival sequences σ. Here $T_{ALG}(\sigma)$ and $T_{OPT}(\sigma)$ denote the throughputs achieved by *ALG* and by an optimal offline algorithm OPT that knows the entire input σ in advance. If *ALG* is a randomized algorithm, then $T_{ALG}(\sigma)$ has to be replaced by the expected throughput $E[T_{ALG}(\sigma)]$.

In practice greedy algorithms are most important. At any time a greedy algorithm serves a queue that currently buffers the largest number of packets. Serving the longest queue is a very reasonable strategy to avoid packet loss if future arrival patterns are unknown. Moreover, greedy strategies are interesting because they are fast and use little extra memory. A switch cannot afford complex computations to decide which queue to serve, nor has it sufficient memory to maintain detailed information on past or current configurations. In this paper we present a thorough study of greedy algorithms and their variants.

Previous work: A simple observation shows that any reasonable algorithm *ALG*, which serves any non-empty queue, is 2-competitive: Partition σ into subsequences σ_l such that *ALG*'s buffers are empty at the end of each σ_l. W.l.o.g. we postpone the beginning of σ_{l+1} until OPT has emptied its buffers, too. If OPT buffers b_i packets in queue i at the end of subsequence σ_l, then at least b_i packets must have arrived there in σ_l. *ALG* has transmitted at least $\sum_{i=1}^{m} b_i$ packets while OPT delivers $\sum_{i=1}^{m} b_i$ more than *ALG* does.

Up to now no deterministic online algorithm with a competitive ratio smaller than 2 has been known. Azar and Richter [3] showed that if $B = 1$, no deterministic strategy can be better than $(2 - \frac{1}{m})$-competitive. For arbitrary B, they gave a lower bound of 1.366. Azar and Richter also considered randomized algorithms and presented a strategy that achieves a competitiveness of $e/(e-1) \approx 1.58$. For $B = 1$, they showed a lower bound of 1.46 on the performance of any randomized online algorithm.

Bar-Noy et al. [5] and Koga [11] studied buffer management policies when buffers have unlimited capacity and one wishes to minimize the maximum queue length. They presented $\Theta(\log m)$-competitive online algorithms. Additional results are known when packets have values and the goal is to maximize the total value of the transmitted packets. Almost all of the previous work has focused on the single queue problem, i.e. we have to maintain only one buffer. Kesselman et al. [7] gave 2-competitive algorithms for various models where preemption is allowed i.e. packets admitted to the queue may be discarded in the event of buffer overflow. Recently, Kesselman et al. [9] developed a 1.983-competitive algorithm when packets must be transmitted in the order they arrive. Aiello et al. [1] investigated single queue problems assuming that preemption is not allowed. For this scenario Andelman et al. [2] showed tight bounds of $\Theta(\log \alpha)$, where α is the ratio of the maximum to the minimum packet value.

Recently Azar and Richter [3] presented a technique that transforms any c-competitive algorithm for a single queue into a $2c$-competitive algorithm for m queues. Using results from [2, 7] they derived 4-competitive preemptive and $2e\lceil \ln \alpha \rceil$- competitive non-preemptive algorithms.

Our contribution: In the first part of the paper we settle the competitive performance of the entire family of greedy algorithms. We prove that a greedy algorithm cannot be better than 2-competitive, no matter how ties are broken. Since any reasonable algorithm is 2-competitive, the competitiveness of any greedy policy is indeed 2. Our lower bound construction is involved and relies on a new recursive construction for building dynamic adversarial buffer configurations. We believe that our technique may be useful for proving lower bounds in other multi-queue buffering problems. In fact, we use a variant of our technique to develop a lower bound for any deterministic online algorithm. We show that, for any buffer size B, no deterministic online strategy *ALG* can achieve a competitiveness smaller than $e/(e-1) \approx 1.58$. Interestingly, we establish this bound by comparing the throughput of *ALG* to that of any greedy algorithm. Using an approach different from [3] we show that for any B, a randomized online algorithm cannot be better than 1.46-competitive.

Although in terms of competitiveness greedy algorithms are not better than arbitrary reasonable algorithms, greedy strategies are important from a practical point of view. Therefore it is interesting to consider variants of greedy policies and to analyze greedy approaches in extended problem settings. In the second part of the paper we develop a slightly modified deterministic greedy stategy, called *Semi-Greedy (SGR)*, and prove that it achieves a competitive ratio of $17/9 \approx 1.89$. We conjecture that *SGR* is actually an optimal deterministic algorithm because for $B = 2$, it achieves an optimal competitiveness of $13/7 \approx 1.86$. These results show, in particular, that deterministic algorithms can beat the factor of 2 and perform better than arbitrary reasonable strategies.

The new *SGR* algorithm is simple. If there is a queue buffering more than $\lfloor B/2 \rfloor$ packets, *SGR* serves a longest queue. If all queues store at most $\lfloor B/2 \rfloor$ packets, *SGR* serves a longest queue that has never buffered B packets provided there is one; otherwise *SGR* serves a longest queue. The idea of this rule is to establish some fairness among the queues. *SGR* is essentially as fast as greedy. It can be implemented such that at most one extra comparison is needed in each time step. The extra memory requirements are also low. For each queue, we just have to maintain one bit indicating whether or not the queue has ever buffered B packets. *SGR* does not follow the standard greedy strategy only if each queue buffers at most $\lfloor B/2 \rfloor$ packets and, hence, the risk of packet loss is low. Thus we consider *SGR* to be a very practical algorithm. We analyze *SGR* by defining a new potential function that measures the number of packets that *SGR* has already lost or could lose if an adversary replenishes corresponding queues. In contrast to standard amortized analysis we do not bound the potential change in each time step. We rather show that if the potential increased at T_1 time steps and $T_1 > C_1$ for some constant C_1, then the potential must have decreased at T_2 steps with $T_2 > C_2$.

In the second part of the paper we also study the case that an online algorithm is granted more resources than an optimal offline algorithm and show that we can beat the competitiveness of 2. We consider resource augmentation with respect to *memory* and *speed*, i.e. we study settings in which an online algorithm has (a) larger buffers in each queue or (b) a higher transmission rate. For scenario (a) we prove that any reasonable algorithm, and in particular any greedy algorithm, achieves a competitive ratio of $(c+2)/(c+1)$ if it has an additional buffer of $A = cB$ in each queue. We show that this bound is tight for greedy strategies. Hence doubling the buffer capacities we obtain a performance ratio of 1.5. For scenario (b) we show an upper bound of $1 + 1/k$ if in each time step an online algorithm can transmit k times as many packets as an adversary. Again, doubling the transmission rate we obtain a competitiveness of 1.5. Finally, we give a linear time offline algorithm for computing an optimal service schedule maximizing the throughput.

This paper is organized as follows. In Section 2 we develop our lower bounds. In Section 3 we present the new *SGR* algorithm and investigate scenarios with resource augmentation. The optimal offline algorithm is given in Section 4.

2 Lower bounds

We first analyze greedy algorithms and then develop lower bounds for arbitrary deterministic and randomized online strategies.

2.1 Greedy Algorithms

Formally, we call an online algorithm GR *greedy* if GR always serves a longest queue.

Theorem 1 *For any B, the competitive ratio of any randomized greedy algorithm GR is not smaller than $2 - 1/B$.*

Proof. Fix a buffer size $B > 0$. We show that there exist infinitely many m and associated packet arrival sequences for m queues such that the throughput achieved by an adversary *ADV* is at least $2 - 1/B - \Theta(m^{-1/2^{B-2}})$ times that achieved by GR. This proves the theorem. We use arrival sequences where whenever there are several queues of maximum lengths, all these queues are served once before the next packets arrive. Thus, the tie-breaking criteria need not be considered.

Let $\mu \geq 2$ be an integer and $b = 2^{B-2}$. Set $m = \mu^b$. We construct a recursive partitioning of the m queues. For any i with $1 \leq i \leq B-2$, let $m_i = m^{1/2^i}$. The m queues are divided into m_1 blocks, each of them consisting of m_1 subsequent queues. These blocks are labeled $1, \ldots, m_1$ in ascending order. Block n_1 with $1 \leq n_1 \leq m_1$ is subdivided into m_2 blocks, each of them consisting of m_2 subsequent queues labeled $(n_1, 1), \ldots, (n_1, m_2)$. This partitioning is repeated up to level $B-2$. In general, any block $(n_1, \ldots, n_i)$ at level i consisting of m_i queues is subdivided into m_{i+1} blocks each containing m_{i+1} queues. These blocks are labeled $(n_1, \ldots, n_i, 1), \ldots, (n_1, \ldots, n_i, m_{i+1})$. Note that a block $(n_1, \ldots, n_{B-2})$ at level $B-2$ consists of exactly μ queues. We define a lexicographic ordering on the $(B-2)$-tuples $(n_1, \ldots, n_{B-2})$ in the standard way. Given $(n_1, \ldots, n_{B-2})$ and $(n'_1, \ldots, n'_{B-2})$ we have $(n_1, \ldots, n_{B-2}) < (n'_1, \ldots, n'_{B-2})$ if $n_i < n'_i$, for some i and $n_j = n'_j$ for all $1 \leq j < i$. Furthermore, $(n_1, \ldots, n_{B-2}) \leq (n'_1, \ldots, n'_{B-2})$ if $(n_1, \ldots, n_{B-2}) < (n'_1, \ldots, n'_{B-2})$ or $n_i = n'_i$ for all $1 \leq i \leq B-2$.

The basic idea of the lower bound construction is to maintain a *staircase of packets* in GR's queues that is centered at some block $(n_1, \ldots, n_{B-2})$. GR's queues in any block $(n'_1, \ldots, n'_{B-2})$ buffer i packets if $n_j = n'_j$, for $1 \leq j \leq i$, but $n_{i+1} \neq n'_{i+1}$. The staircase center moves through the blocks in increasing lexicographic order. When the center is located at $(n_1, \ldots, n_{B-2})$ we force a packet loss of B at each of GR's queues in that block. *ADV* will be able to accept all packets and essentially has full queues in all blocks $(n'_1, \ldots, n'_{B-2})$ that are lexicographically smaller than $(n_1, \ldots, n_{B-2})$. When the construction ends, almost all of *ADV*'s queues are fully populated while GR's queues are empty. Since a total of nearly mB packets are transmitted by *ADV* and GR during the construction, this gives the desired lower bound.

Formally, we process blocks $(n_1, \ldots, n_{B-2})$ with $n_i \geq 2$ for all i in increasing lexicographic order. Blocks $(n_1, \ldots, n_{B-2})$ with $n_i = 1$ for some i are special in that less than B packets will arrive there. When we start processing a block $(n_1, \ldots, n_{B-2})$ with $n_i \geq 2$ for all i, certain invariants given below hold. We show how to process it such that the invariants are also when we start processing the next block.

(G1) Let $(n'_1, \ldots, n'_{B-2})$ be a block with $(n'_1, \ldots, n'_{B-2}) < n_1, \ldots, n_{B-2})$ and $n'_i \geq 2$ for all i. GR buffers exactly $j+1$ packets if j is the largest index with $n'_1 = n_1, \ldots, n'_j = n_j$.

(G2) Let $(n'_1, \ldots, n'_{B-2})$ be a block with $(n'_1, \ldots, n'_{B-2}) < (n_1, \ldots, n_{B-2})$ such that $n'_i = 1$ and $n'_j \geq 2$ for all $1 \leq j \leq i-1$. GR has $j+1$ packets in each of the queues if $n'_j = n_j$ for all $1 \leq j \leq i-1$.

(G3) Let $(n'_1, \ldots, n'_{B-2})$ be a block with $(n'_1, \ldots, n'_{B-2}) \geq (n_1, \ldots, n_{B-2})$. GR buffers exactly j packets if j is the largest index with $n'_1 = n_1, \ldots, n'_j = n_j$.

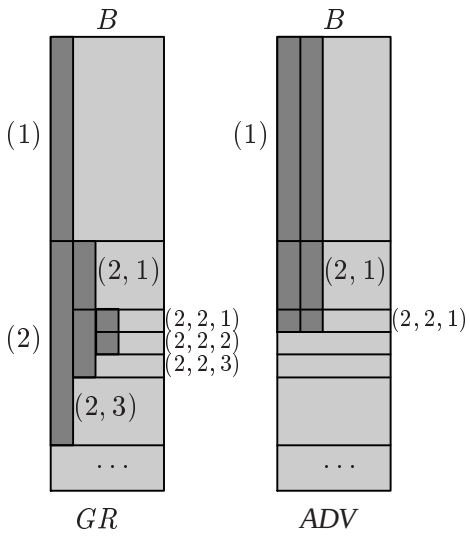

Figure 1: Queue configurations when we start processing block $(2, 2, 2)$.

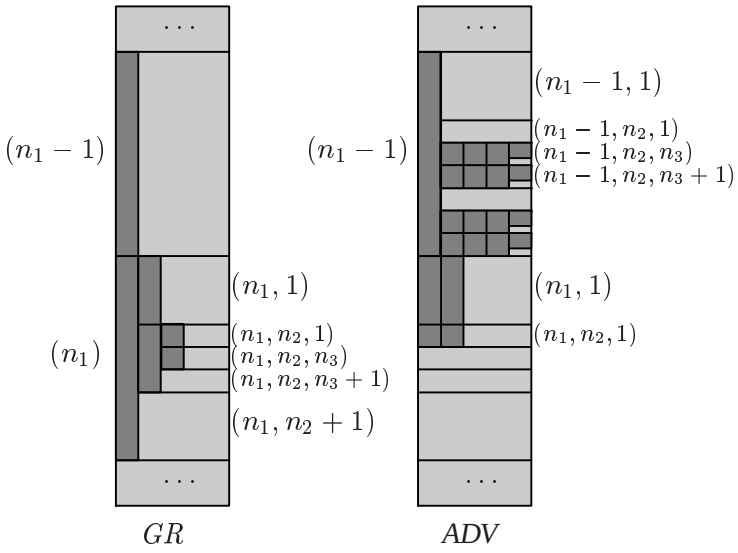

Figure 2: Queue configurations when we start processing block (n_1, n_2, n_3).

(A1) Let $(n'_1, \ldots, n'_{B-2})$ be a block with $(n'_1, \ldots, n'_{B-2}) < (n_1, \ldots, n_{B-2})$ and $n'_i \geq 2$ for all i. *ADV* has B packets in each of the first $m_{B-2} - 1 = \mu - 1$ queues and $B - 1$ packets in the last queue of this block.

(A2) Let $(n'_1, \ldots, n'_{B-2})$ be a block with $(n'_1, \ldots, n'_{B-2}) < (n_1, \ldots, n_{B-2})$ such that $n'_i = 1$ and $n'_j \geq 2$ for all $1 \leq j \leq i - 1$. *ADV* has two packets in each of the queues if $n'_j = n_j$ for all $1 \leq j \leq i - 1$ and one packet in those queues otherwise.

(A3) Let $(n'_1, \ldots, n'_{B-2})$ be a block with $(n'_1, \ldots, n'_{B-2}) \geq (n_1, \ldots, n_{B-2})$. *ADV* has 0 packets in each of those queues.

Initialization: We show how to establish the six invariants for the block $(2, \ldots, 2)$. At the beginning there arrive $2m_1$ packets in the queues of block (1) at level 1, two packets in each queue, and m_1 packets in the queues of block (2) at level 1, one packet in each queue. GR starts transmission from block 1 while *ADV* does so from block 2. After both GR and *ADV* have transmitted m_1 packets, we jump to level 2 where the arrival pattern is adopted in gauge: there arrive $2m_2$ packets in block $(2, 1)$, two in each queue, and m_2 packets in block $(2, 2)$, one in each queue. We continue to copy and scale down this pattern until blocks $(2, \ldots, 2, 1)$ and $(2, \ldots, 2, 2)$ at level $B - 2$ are reached. At level i, GR always clears m_i packets in block $(2, \ldots, 2, 1)$ while *ADV* does so in $(2, \ldots, 2, 2)$. Figure 1 shows these initial configurations.

Invariants (A1) and (G1) trivially hold because there is no block $N' < (2, \ldots, 2)$ with $n'_i \geq 2$, for all $1 \leq i \leq B - 1$. If $N' < (2, \ldots, 2)$ and j is the smallest index with $n'_j = 1$, then $n'_1 = \ldots = n'_{j-1} = 2$. Queues in N' received $j + 1$ packets, $j - 1$ of which have been transmitted by *ADV*, while only one of them has been transmitted by GR. Hence, invariants (A2) and (G2) hold. If $N' \geq (2, \ldots, 2)$ and j is the largest index with $n'_1 = n_1, \ldots, n'_j = n_j$, then queues in N' received j packets, all of which have been transmitted by *ADV*, while none of them have been transmitted by GR, giving that invariants (A3) and (G3) hold.

Processing of a block: We next describe how to process block $N = (n_1, \ldots, n_{B-2})$. Figure 2 shows the buffer configurations when the processing starts. Let $q_1, \ldots, q_{m_{B-2}}$ be the m_{B-2} queues in that block. By (G3), GR has $B - 2$ packets in each of these queues. By (A3), *ADV* stores no packets there. We subsequently apply the following arrival pattern to each but the last of the q_j's: There arrive B packets at

queue q_j and one packet at queue q_{j+1}. First, GR accepts two packets in q_1 and one packet in q_2. Then q_1 is completely populated while q_2 still has one free buffer left. Afterwards GR transmits a packet from q_1. At the arrival of B packets at q_2, GR must reject all but one of them and can accept the additional packet at q_3 as well. This behavior is repeated until the last queue in N is reached. In contrast, *ADV* always processes the single packet in q_{j+1} in order to be able to accept B packets in the next step. When B packets arrive at $q_{m_{B-2}}$, we leave the additional packet out because we would cross the boundary to the next block of level $B-2$. We assume that both *ADV* and GR then transmit one packet from $q_{m_{B-2}}$. Hence GR stores $B-1$ packets in each queue of N, whereas *ADV* buffers B packets in all but the last queue and $B-1$ packets in the last one. We next show how to establish the six invariants for the next block $\overline{N} = (\overline{n}_1, \ldots, \overline{n}_{B-2})$ in the lexicographic order satisfying $\overline{n}_i \geq 2$, for all i. We distinguish cases depending on whether or not $\overline{n}_{B-2} = n_{B-2} + 1$.

Case 1: If $\overline{n}_{B-2} = n_{B-2} + 1$, then N and $\overline{N}$ belong to the same block of level $B-3$. By (G3), GR buffers $B-3$ packets in each queue of $\overline{N}$. Now there arrive m_{B-2} packets at $\overline{N}$, one at each of the queues. Thus, each queue in N buffers $B-1$ packets, and each queue in $\overline{N}$ buffers $B-2$ ones. In the following m_{B-2} time steps GR transmits one packet from each queue in N while *ADV* serves the queues in $\overline{N}$, which are then empty again in *ADV*'s configuration. Since only N and $\overline{N}$ have been affected, the invariants (G1), (G2) and (G3) hold for $\overline{N}$ as well. The same is true for (A1) because the statement holds for block N, as argued at the end of the last paragraph. (A2) was not affected during the processing of N because $\overline{n}_i \geq 2$, for all $1 \leq i \leq B-1$. Since no new packets arrived at blocks that are lexicographically larger than $\overline{N}$, (A3) is also satisfied.

Case 2: If $\overline{n}_{B-2} \neq n_{B-2} + 1$, then N and $\overline{N}$ do not belong to the same block at level $B-3$. Let i be the largest index such that $n_i < m_i$, i.e. there is another block at level i. Hence $\overline{N} = (n_1, \ldots, n_{i-1}, n_i + 1, 2, \ldots, 2)$. In the following time steps, no new packets arrive, and since (G1) and (G2) hold, GR transmits m_j packets from the queues of block $(n_1, \ldots, n_i, m_{i+1}, \ldots, m_j)$, for $j = B-2, \ldots, i+1$. During each iteration, one packet is transmitted from each of these queues. In these time steps, for $j = B-2, \ldots, i+1$, *ADV* transmits one packet from each of the queues in $(n_1, \ldots, n_i, m_{i+1}, \ldots, m_{j-1}, 1)$. By invariant (A2), these queues hold exactly two packets in *ADV*'s configuration and store precisely one packet after the transmission. In the next time step m_i packets arrive at the queues of $(n_1, \ldots, n_i + 1)$, one packet at each of these queues. At that time in GR's configuration, the queues in $(n_1, \ldots, n_i)$ buffer exactly $i+1$ packets while all other queues buffer less. GR then transmits one packet from each of the queues in $(n_1, \ldots, n_i)$ while *ADV* serves the queues in $(n_1, \ldots, n_i + 1)$ so that they are empty again. In the following steps we restore in GR's configuration the full staircase on top of the i packets in the queues of $(n_1, \ldots, n_i + 1)$. More precisely, there arrive $2m_{i+1}$ packets at the queues of $(n_1, \ldots, n_i + 1, 1)$, two at each of these queues, and m_{i+1} packets at the queues of $(n_1, \ldots, n_i + 1, 2)$, one packet at each of these queues. GR transmits one packet from each queue in $(n_1, \ldots, n_i + 1, 1)$ while *ADV* clears block $(n_1, \ldots, n_i + 1, 2)$. Then there arrive $2m_{i+2}$ packets in $(n_1, \ldots, n_i + 1, 2, 1)$ and m_{i+2} packets in $(n_1, \ldots, n_i + 1, 2, 2)$. Again GR serves the queues in the first of these blocks while *ADV* clears the second. This process continues up to blocks $(n_1, \ldots, n_i + 1, 2, \ldots, 2, 1)$ and $(n_1, \ldots, n_i + 1, 2, \ldots, 2)$ at level $B-2$.

Lemma 1 *Invariants (G1–3) and (A1–3) hold when we start processing* $\overline{N} = (n_1, \ldots, n_i + 1, 2, \ldots, 2)$.

Proof. Consider block $\overline{N} = (n_1, \ldots, n_i + 1, 2, \ldots, 2) =: (\overline{n}_1, \ldots, \overline{n}_{B-2})$. Let $N' = (n'_1, \ldots, n'_{B-2})$ be an arbitrary block. We first study (G1). Let $N' < N$, $n'_j \geq 2$ for all j, and let k be the largest index such that $n'_1 = n_1, \ldots, n'_k = n_k$. If $k < i$, then there is nothing to show because the queues in N' have not been touched by GR since the processing of N started. If $k \geq i$, we have $N' = (n_1, \ldots, n_i, m_{i+1}, \ldots, m_k, n'_{k+1}, \ldots, n'_{B-2})$. So N' is affected at the iteration steps for $j = k, \ldots, i$, hence $k - i + 1$ times, where iteration i corresponds to the subsequent transmission of one packet from each queue in $(n_1, \ldots, n_i)$. Since N' buffered $k+1$ packets before the processing of N started, there are $k + 1 - (k - i + 1) = i$ packets after the iteration steps. On the other hand, $i - 1 = \max\{j : n'_1 = \overline{n}_1, \ldots, n'_j = \overline{n}_j\}$. For $N' = N$ the

statement of (G1) also holds because exactly i packets are buffered at these queues. If $N' < N$, statement (G2) holds because of the same arguments, starting with k packets before the processing and eventually getting i packets.

If $N < N' < \overline{N}$, then let j be the largest index with $n'_1 = \overline{n}_1, \ldots, n'_j = \overline{n}_j$. We have $i \leq j < B-2$ and $n'_{j+1} = 1$. Since $n'_{i+1} = \ldots = n'_j = 2$, GR buffers exactly $j+1$ packets in the queues of N'. Hence (G2) holds here as well. Moreover, since GR has $B-2$ packets in the queues of $\overline{N}$, (G3) holds for $N' = \overline{N}$. If $N' > \overline{N}$, then we distinguish two cases. If $n'_l > \overline{n}_l$ for some $1 \leq l \leq i$, there is nothing to show because GR's configuration in these queues has not changed and the largest index j with $n_1 = n'_1, \ldots, n_j = n'_j$ is equal to the largest index j with $\overline{n}_1 = n'_1, \ldots, \overline{n}_j = n'_j$. If $n'_1 = \overline{n}_1, \ldots, n'_i = \overline{n}_i = n_i + 1$, then let j be the largest index with $n'_1 = \overline{n}_1, \ldots, n'_j = \overline{n}_j$. We have $n'_{i+1} = 2, \ldots, n'_j = 2$ and $n'_{j+1} > 2$. Hence the queues in N' store exactly j packets and (G3) holds.

Invariant (A1) obviously holds because it held when we started processing N and the desired property was established for block N. There exist no blocks N' with $N < N' < \overline{N}$ and $n'_i \geq 2$ for all i. Invariant (A2) is satisfied for blocks N' with $N' \leq N$ because during the processing of N, *ADV* served the queues in$(n_1, \ldots, n_i, m_{i+1}, \ldots, m_{j-1}, 1)$ for $j = B-1, \ldots, i+1$ exactly once, thus reducing the number of packets stored there from 2 to 1. If $N' > N$, then the queues store exactly two packets, as desired. Finally, (A3) holds because *ADV* has transmitted all packets that have arrived at blocks $N' \geq \overline{N}$. □

After processing the last block $(m_1, \ldots, m_{B-2})$, no further packets arrive, but the iteration steps are executed as if there were another block. Then, we have the following configurations: From invariants (G1), (G2) and (G3), we derive that GR buffers one packet in each queue. Due to (A1), (A2) and (A3), *ADV* buffers $B-1$ or B packets in the queues in blocks $(n_1, \ldots, n_{B-2})$ with $n_i \geq 2$ for all i while the others buffer exactly one packet like GR does.

Let T_G be the throughput achieved by GR and T_A be the throughput achieved by *ADV*. For any block $(n_1, \ldots, n_{B-2})$ with $n_i \geq 2$ for all i, GR transmits B packets from each of the associated queues. For any block $(n_1, \ldots, n_{i-1}, 1)$ with $n_j \geq 2$, for $j = 1, \ldots, i-1$, GR transmits $i+1$ packets from each queue. There are $\prod_{j=1}^{i-1}(m_j - 1)$ such blocks, each containing m_i queues. Thus $T_G = (m - \breve{m})B + \delta_1$, where

$$\breve{m} = \sum_{i=1}^{B-2} \prod_{j=1}^{i-1} (m_j - 1) m_i \quad \delta_1 = \sum_{i=1}^{B-2} \prod_{j=1}^{i-1} (m_j - 1) m_i (i+1).$$

The throughput of *ADV* is equal to that of GR plus the number of packets that are in *ADV*'s final configuration when the processing of blocks ends minus the number of packets that are in GR's final configuration when the processing of blocks ends. In this final configuration queues in blocks $(n_1, \ldots, n_{B-2})$ with $n_i \geq 2$ for all i store B packets, except for the last of these queues which buffers only $B-1$ packets. All other queues are empty. Hence $T_A = T_G + (m - \breve{m})B - \delta_2 - (m - \breve{m})$, where $\delta_2 = (m - \breve{m})/\mu$. Hence $T_A \geq T_G + (m - \breve{m})(B-1) - m\mu^{-1}$. Moreover $\breve{m} \in \Theta(\prod_{j=1}^{B-2} m_j) = \Theta(\prod_{j=1}^{B-2} m^{\frac{1}{2^j}}) = \Theta(m^{1-\frac{1}{2^{B-2}}})$, $m\mu^{-1} \in \Theta(m^{1-\frac{1}{2^{B-2}}})$, and $\delta_1 \in \Theta(B \prod_{j=1}^{B-2} m_j) = \Theta(B m^{1-\frac{1}{2^{B-2}}})$. This implies

$$\begin{aligned}
\frac{T_A}{T_G} &\geq \frac{(m - \breve{m})B + \delta_1 + (m - \breve{m})(B-1) - m\mu^{-1}}{(m - \breve{m})B + \delta_1} = 2 - \frac{(m - \breve{m}) + \delta_1 + m\mu^{-1}}{(m - \breve{m})B + \delta_1} \\
&\geq 2 - \frac{1}{B} - \frac{\delta_1 + m\mu^{-1}}{(m - \breve{m})B + \delta_1} \in 2 - \frac{1}{B} - \Theta\left(\frac{B m^{1-\frac{1}{2^{B-2}}}}{mB}\right) = 2 - \frac{1}{B} - \Theta\left(m^{-\frac{1}{2^{B-2}}}\right).
\end{aligned}$$

□

2.2 Arbitrary Algorithms

We first study deterministic online algorithms and then address randomized strategies.

Theorem 2 *The competitive ratio of any deterministic online algorithm ALG is at least* $e/(e-1)$.

Proof. Let B be a positive integer representing the buffer size. For any positive integer N, let $m = (B+1)^N$ be the number of queues. Let the B buffer columns be indexed $1, \ldots, B$ where column B is the one at the head of the queues and column 1 is the one at the tails. At the beginning there arrive B packets at each of the m queues such that all buffers are fully populated. We present a scheme S for constructing request sequences σ. Our scheme has the property that the throughput achieved by an adversary is at least $e/(e-1)$ times the throughput obtained by any greedy algorithm and that the throughput of any greedy strategy is at least as large as the throughput of any deterministic online algorithm *ALG*. This establishes the theorem. It will be sufficient to consider the standard greedy algorithm, denoted by GR, which serves the smallest indexed queue in case there is more than one queue of maximum length.

A request sequence σ consists of superphases $P_1, \ldots, P_B$. Superphase P_i consists of phases $(i, N), \ldots, (i, 1)$. In a phase (i, s) packets arrive at the $q_s = (B+1)^{s-1}$ most populated queues in the online configuration. We first present an informal description of the superphases and phases, explaining how GR would serve them. Then we give a formal definition with respect to any online strategy. When superphase P_i is finished, the last i columns in GR's buffers are empty while the other columns are fully populated. In particular, when σ ends, all buffers are empty. Superphase P_i is meant to empty column i. After phase (i, s) the first $m - q_s$ queues contain $B - i$ packets while the remaining queues buffer $B - i + 1$ packets.

We first describe superphase P_1 and start with phase $(1, N)$. Initially, all $q_{N+1} = m$ queues are fully populated. During the first $q_N B$ time steps no packets arrive and GR transmits a packet from each of the first $q_N B$ queues. Then B packets arrive at each of the remaining q_N queues, all of which must be dropped by GR because the last q_N queues already buffer B packets each. An adversary, on the other hand, could transmit all packets from the last q_N queues during the first $q_N B$ time steps so that no packet loss occurs. At the end of phase $(1, N)$ the last q_N queues are fully populated in GR's buffer configuration. The arrival pattern now repeats for the other phases in P_1. At the beginning of $(1, s)$ the last q_{s+1} queues in GR's configuration store B packets each. During the first $q_s B$ time steps no packets arrive and GR transmits one packet from each of the first $q_s B$ of these q_{s+1} queues. Then B packets are sent to each of the last q_s queues, all of which are lost by GR. Again, an adversary can accept all packets by transmitting from the last q_s queues in the previous time steps. At the end of $(1, 1)$ the last queue in GR's configuration contains B packets while all other queues buffer $B - 1$ packets. Now there is one time step without packet arrivals such that GR can transmit one packet from the last queue and has exactly $B - 1$ packets in each of its buffers.

Superphase P_2 is similar. In $(2, N)$ there are no packet arrivals during the first $q_N B$ time units. Then there arrive B packets at each of the last q_N queues. This time GR loses $B - 1$ packets at each of the last queues, which are then fully populated again. To these last q_N queues we recursively apply the packet arrival pattern used to empty column 1 in P_1. In phase $(2, s)$ there are no packet arrivals during the first $q_s B$ time units. Then B packets are sent to the last q_s queues, causing a packet loss of $(B-1)$ at each of these buffers. To empty the last column of these queues, we recursively apply the pattern of P_1. In general, in any phase (i, s) of P_i there are $q_s B$ time units without packet arrivals, followed by the arrival of B packets at each of the last q_s queues. GR loses $B - i + 1$ packets at each of these buffers, which are then fully populated again. To empty the last $i - 1$ columns of these buffers we recursively apply the pattern used in P_{i-1}.

Formally, for any deterministic online algorithm *ALG*, $\sigma = P_1, \ldots, P_B$, where $P_i = (i, N), \ldots, (i, 1)$. We call the BN phases in $P_1, \ldots, P_B$ *main phases*. A main phase triggers *auxiliary phases*. Auxiliary phases are identified by their complete triggering path. If φ_1 is a main phase and φ_i triggers φ_{i+1}, then the auxiliary phase φ_n is denoted by $[\varphi_1 \varphi_2 \ldots \varphi_n]$. Let Φ be the set of phases. To identify sets of queues on which phases work, we define a predecessor mapping $\pi : \Phi \to \Phi$. For a main phase $\varphi = (i, s)$, let i be the level of φ. If φ is a main phase, then $\pi(\varphi)$ is the immediately preceding phase in σ of the same level if such exists; otherwise we define $\pi(\varphi) = (0, 0)$. If φ is an auxiliary phase, then $\pi(\varphi)$ is the triggering phase.

- $\pi((i,N)) = (0,0)$ and $\pi((i,s)) = (i,s+1)$ if $s < N$
- $\pi([(i_1,s_1)\dots(i_n,s_n)])$ = $[(i_1,s_1)\dots(i_{n-1},s_{n-1})]$ if $s_n = s_{n-1} - 1$
- $\pi([(i_1,s_1)\dots(i_n,s_n)])$ = $[(i_s,s_1)\dots(i_n,s_n+1)]$ if $s_n < s_{n-1} - 1$

Furthermore we need a recursive mapping $Q : \Phi \to \mathcal{P}(\{1,\dots,m\})$ that associates phases with sets of queues. Let $Q(0,0)$ be the set of all m queues.

Each phase φ with suffix (i,s) consists of the following steps.

1. $q_s B$ time steps without packet arrival, i.e. only packet transmission.
2. Arrival of B packets at each of the q_s most populated queues in $Q(\pi(\varphi))$; these queues form $Q(\varphi)$.
3. If $i > 1$ and $s > 1$, triggering of the phases $[\varphi,(1,s-1)],\dots,[\varphi,(1,1)],\dots,$ $[\varphi,(i-1,s-1)],\dots[\varphi,(i-1,1)]$ on $Q(\varphi)$.
4. If $s = 1$, then i time steps without packet arrival.

Lemma 2 *If a sequence of phases $(1,s),\dots,(1,1),\dots,(i,s),\dots,(i,1)$ is served by GR on a set $Q_{G,s}$ consisting of $q_{s+1} = (B+1)^s$ consecutive queues, then after phase (j,r) the buffer configuration in $Q_{G,s}$ is as follows. If $r > 1$, then the last q_r queues buffer $B - j + 1$ packets while the other queues buffer $B - j$ packets. If $r = 1$, then all queues buffer $B - j$ packets.*

Proof. We first prove the statement of the lemma under the condition that queues not contained in $Q_{G,s}$ buffer less than $B - i$ packets. Then we show that the condition is always satisfied.

The proof is by induction on i. First consider $i = 1$. At the beginning all buffers in $Q_{G,s}$ are full because the sequence is either a sequence of main phases started with fully populated buffers or a sequence of triggered phases which are also initiated on full buffers only. In $(1,s)$ during the first $q_s B$ time steps, GR transmits a packet from the first $q_s B$ queues in $Q_{G,s}$ because queues outside $Q_{G,s}$ contain less than B packets. In the next time step the packets arriving at the last q_s queues in $Q_{G,s}$ are rejected because the queues already buffer B packets. No further phases are triggered in $(1,s)$. Thus the last q_s queues buffer B packets while the other queues store $B - 1$ packets. If $s = 1$, then $q_s = 1$ and in the last time step in $(1,1)$ a packet is transmitted from the last queue so that all queues buffer exactly $B - 1$ packets. The desired statement on the buffer population holds after phase $(1,s)$. Suppose inductively that it holds after $(1,r+1)$. Then the last q_{r+1} queues in $Q_{G,s}$ store B packets while the remaining queues each have $B - 1$ packets. During the next $q_r B$ time steps in $(1,r)$ one packet is transmitted from the first $q_r B$ queues among the last q_{r+1} buffers in $Q_{G,s}$. Thus the last q_r queues in $Q_{G,s}$ still have B packets while the remaining queues buffer $B - 1$ packets. Again, if $r = 1$, one packet is transmitted from the last queue in the final time steps in $(1,r)$ so that all queues have exactly $B - 1$ packets. Thus the stated buffer population is maintained after phase $(1,r)$ and the desired statement holds for $i = 1$.

Suppose that the statement holds for integers smaller than i; we prove that it is also satisfied for i. As above we can show that after phase $(1,s)$ the buffer population is as desired. Assume that the statement on the buffer population holds up to but excluding phase (j,r). At the beginning of (j,r) the last q_{r+1} buffers in $Q_{G,s}$ store $B - j + 1$ packets while the remaining queues contain $B - j$ packets. During the next time steps, GR transmits one packet from each of the first $q_r B$ buffers among the q_{r+1} last ones in $Q_{G,s}$ because buffers outside $Q_{G,s}$ store less than $B - i < B - j + 1$ packets. Thus the last q_r queues in $Q_{G,s}$ store $B - j + 1$ packets while the remaining queues each contain $B - j$ packets. Then B packets arrive at the last q_r queues so that they are fully populated again. If $r = 1$, then no phases are triggered and since $q_1 = 1$, only the last queue is fully populated. During the next j time steps j packets are transmitted from this last queue and all queues in $Q_{G,s}$ then buffer $B - j$ packets. The queues are populated as desired. If $r > 1$, then

phases $(1, r-1), \ldots, (1,1), \ldots, (j-1, r-1), \ldots, (j-1, 1)$ are triggered on the last q_r fully populated queues. The other queues buffer less than $B-(j-1)$ packets. By induction hypothesis when the triggered phases end, the last q_r queues buffer again $B-j+1$ packets. Thus the buffers are populated as desired and the desired statement holds for integers i.

It remains to show that the condition mentioned at the beginning of the proof holds, i.e. if a sequence of phases $(1,s), \ldots, (1,1), \ldots, (i,s), \ldots, (i,1)$ is served on a set $Q_{G,s}$ of queues, then all queues not contained in $Q_{G,s}$ store less than $B-i$ packets. The condition holds when the initial sequence σ of main phases is started. Suppose that the condition holds for all sequences triggered after the beginning of σ but before the start of $(1,s), \ldots, (1,1), \ldots, (i,s), \ldots, (i,1)$. Assume that the latter sequence is triggered by phase $[\varphi_1 \ldots \varphi_n]$ such that $\varphi_j = (i_j, s_j)$ and phase φ_j is executed on queues in Q_{φ_j}. We have $Q_{\varphi_j+1} \subset Q_{\varphi_j}$ for $j = 1, \ldots, n-1$, and queues outside Q_{φ_j} contain less than $B-i_j$ packets. Thus all queues outside Q_{φ_n} buffer less than $B-i_n$ packets. The sequence is triggered on the last q_{s_n} buffers in Q_{φ_n}. Since the first $q_{s_n}B$ buffers in Q_{φ_n} store $B-i_n$ packets and $i_1 > i_2 > \ldots > i_n > i$, the sequence is triggered on a set of buffers storing less than $B-i$ packets. The proof is complete. □

Lemma 3 *The packet loss of any deterministic online algorithm ALG is not smaller than the one of GR.*

Proof. To establish the claim, we use Lemma 2. We consider sequences $\sigma(i,s) = (1,s), \ldots, (1,1), \ldots, (i,s), \ldots, (i,1)$ processed by *ALG* and GR on sets Q_A and Q_G of q_s buffers each. For any given time since the beginning of $\sigma(i,s)$ let N_G be the total number of packets in GR's queues of Q_G and let L_G be the total packet loss of GR accumulated since the start of $\sigma(i,s)$. For algorithm *ALG*, N_A and L_A are defined similarly. Furthermore let S_A be the number of time steps in $\sigma(i,s)$, where neither packets arrive nor does *ALG* transmit a packet from queues in Q_A. We are interested in the following invariants.

$$\text{(I1)}\ \ N_A - S_A + L_A = N_G + L_G \qquad \text{(I2)}\ \ N_A - S_A \leq N_G$$

We prove the following statement. If a sequence $\sigma(i,s)$ is processed by *ALG* and GR on sets Q_A and Q_G, respectively, then after each phase invariants (I1) and (I2) hold. The lemma then follows by considering sequence $\sigma = \sigma(B,N)$. At the end of the sequence, by (I1), $L_A = N_G + L_G - (N_A - S_A)$. By (I2) we conclude $L_A \geq L_G$.

We prove the desired statement by induction on i. First consider $i = 1$. At the beginning of $\sigma(1,s)$ all queues in Q_A and Q_G are fully populated because sequence σ as well as triggered phases are initiated on full buffers only. Therefore $N_A = N_G$. Since initially $L_A = L_G = 0$ and $S_A = 0$, both invariants hold at the beginning of $\sigma(1,s)$. While serving $\sigma(1,s)$, in each time step GR transmits a packet from queues in Q_G. Moreover, GR always serves a fully populated queue. Therefore, at any time, GR always has the smallest number of fully populated queues. By Lemma 2, at the beginning of phase $(1,r)$ the last q_{r+1} queues in Q_G buffer B packets while all other queues store $B-1$ packets. After the next q_rB time steps the last q_r queues still buffer B packets such that all packets arriving in step 2 of $\sigma(1,r)$ are lost. Since *ALG* has at least q_r fully populated queues, *ALG* also loses all incoming packets. Therefore, at any time the packet loss of *ALG* and GR is the same. Since GR transmits a maximum number of packets and no incoming packets can be accepted by *ALG* and GR, $N_A \geq N_G$. If $N_A - N_G = \delta > 0$, then there must have been δ time units where *ALG* did not transmit a packet. Therefore (I1) and (I2) hold throughout the execution of $\sigma(1,s)$.

Assume that the statement to be proven holds for integers smaller than i. We consider a sequence $\sigma(i,s)$. Again (I1) and (I2) hold initially, when $\sigma(i,s)$ is started. We show that if the invariants (I1) and (I2) hold before any phase (j,r), then they also hold after the phase. At the beginning of (j,r), GR buffers $B-j+1$ packets in the last q_{r+1} queues of Q_G and $B-j$ packets in the remaining queues. During the first q_rB time steps, GR always transmits one packet. Consider any time step. If *ALG* also transmits a packet from queues in Q_A, then both N_A and N_G decrease by 1. If *ALG* does not serve a queue in Q_A, then only N_G decreases by 1 but S_A increases by 1. The invariants are maintained. Immediately before step 2 of

phase (j,r) let Q_A^1 be the set of the q_r most populated buffers in Q_A to which packets are sent in step 2. Set $Q_A^2 = Q_A \setminus Q_A^1$ and let N_A^i be the number of packets in $Q_A^i, i = 1,2$. Similarly, let Q_G^1 be the set of the q_r most populated queues in Q_G, $Q_G^2 = Q_G \setminus Q_G^1$ and N_G^i be the packet number in $Q_G^i, i = 1,2$. Each queue in Q_G^2 has exactly $B-j$ packets while each queue in Q_G^1 buffers exactly $B-j+1$ packets. We have $N_A^2 - S_A \leq N_G^2$. Otherwise, if $N_A^2 - S_A > N_G^2$, then the most populated queues in Q_A^2 would store at least $B-j+1$ packets. Hence $N_A^1 \geq N_G^1$ and (I1) would be violated.

Now consider the effect of step 2 of phase (j,r) in which the queues of Q_A^1 and Q_G^1 are refilled. If $L_A - L_G$ changes by δ, then $N_G - N_A$ changes by δ so that (I1) is maintained. Since $N_A^1 = N_G^1$ after the step and $N_A^2 - S_A \leq N_G^2$, invariant (I2) also holds. If $r = 1$, then no phases are triggered and there are j more time steps with packet transmission in which the invariants are maintained. If $r > 1$, then phases $(1,r-1),\ldots,(1,1),\ldots,(j-1,r-1),\ldots,(j-1,1)$ are triggered in step 3 of (j,r). The triggered phases are executed on Q_A^1 and Q_G^1. Since $N_A^1 = N_G^1$ initially, the desired invariants hold at the beginning of the triggered sequence. By induction hypothesis we have $N_A^1 - S_A^1 + L_A^1 = N_G^1 + L_G^1$ and $N_A^1 - S_A^1 \leq N_G^1$ when the sequence is finished. Here L_A^1 and L_G^1 denote the loss incurred during the sequence and S_A^1 is the number of time steps, *ALG* does not transmit packets from Q_A^1. Let $S_A^{1,1}$ be the number of time steps in the triggered sequence where *ALG* works neither on Q_A^1 nor on Q_A^2 and let $S_A^{1,2}$ be the number of time steps where *ALG* does not work on Q_A^1 but does transmit packets from Q_A^2. Then $S_A^1 = S_A^{1,1} + S_A^{1,2}$. Before step 3 of the phase (I1) held and $N_A^1 = N_G^1$. Therefore $N_A^2 - S_A + L_A = N_G^2 + L_G$ and, as argued above, $N_A^2 - S_A \leq N_G^2$. Using the invariants for the triggered sequence we obtain

$$(N_A^1 + N_A^2 - S_A^{1,2}) - (S_A + S_A^{1,1}) + (L_A + L_A^1) = (N_G^1 + N_G^2) + (L_G + L_G^1)$$

and

$$(N_A^1 + N_A^2 - S_A^{1,2}) - (S_A + S_A^{1,1}) \leq (N_G^1 + N_G^2).$$

The invariants thus hold at the end of phase (j,r) because $N_A^1 + N_A^2 - S_A^{1,2}$ is the total number of packets in queues from Q_A, $S_A + S_A^{1,1}$ is the total number of time steps *ALG* does not transmit packets from Q_A and $N_G^1 + N_G^2$ is the number of packets in Q_G. Finally, $L_A + L_A^1$ is the total loss accumulated by *ALG* and $L_G + L_G^1$ is the corresponding value of GR. The inductive proof is complete. □

Lemma 4 *The throughput of the adversary ADV is at least $e/(e-1)$ times that of GR.*

Proof. We analyze the competitive ratio of the GR algorithm. To this end we analyze the throughput T_{ADV} achieved by an adversary ADV and the throughput T_G of GR. At the beginning of σ, all queues are fully populated. When σ ends, by Lemma 2, all of GR's buffers are empty. We describe how ADV serves σ. In phase (i,s) during the first $q_s B$ time steps *ADV* transmits all packets from the q_s queues at which packets arrive in step 2 of the phase. If $s = 1$, then in step 4 of the phase *ADV* is idle and does not transmit packets. Thus in each phase *ADV* can always accept all incoming packets and at the end of the phase all buffers are full. Let S_{ADV} the total number of time steps where *ADV* is idle in σ. Since GR transmits a packet in each time step and *ADV*'s queues contain mB packets at the end of σ, we have $T_G = T_{ADV} - mB + S_{ADV}$.

We estimate S_{ADV}. The adversary is idle in a phase $\varphi = [\varphi_1 \ldots \varphi_n]$ with $\varphi_j = (i_j, s_j)$ if $s_n = 1$. Since $i_1 > \ldots > i_n$ and $i_j > 1$ for $j < n$, there are $\sum_{n=1}^{B} \binom{B}{n}(N-1)^{n-1} \leq N^B$ such phases, in each of which *ADV* is idle for at most B time units. Thus $S_{ADV} \leq BN^B$ and the competitive ratio of GR is at least

$$c \geq T_{ADV}/T_G \geq 1/\left(1 - \frac{mB}{T_{ADV}} + \frac{N^B B}{T_{ADV}}\right) \geq 1/\left(1 - \frac{mB}{T_{ADV}} + \frac{N^B}{m}\right)$$

because $T_{ADV} \geq mB$. To lower bound the last expression we have to upper bound T_{ADV}. The throughput of *ADV* is equal to the mB packets initially buffered in the queues plus the packets accepted by ADV during the service of σ. As argued above, *ADV* can always accept all packets. Let $a_{i,s}$ be the number of packets

accepted by *ADV* in a phase (i,s). We show by induction on i that $a_{i,s} \leq (B+1)^s(\frac{B+1}{B})^{i-2}$. The inequality holds for $i=1$ because $a_{1,s} = q_sB = (B+1)^{s-1}B$. Suppose that the inequality holds for i. We have $a_{i+1,s} = q_sB + \sum_{j=1}^{i}\sum_{r=1}^{s-1} a_{j,r} = a_{i,s} + a_{i,s-1} + \ldots + a_{i,1} \leq (\frac{B+1}{B})^{i-2}\sum_{r=1}^{s}(B+1)^r \leq (\frac{B+1}{B})^{i-1}(B+1)^s$. The total number of accepted packets in σ is at most

$$\begin{aligned}\sum_{i=1}^{B}\sum_{s=1}^{N} a_{i,s} &\leq \sum_{i=1}^{B}(\frac{B+1}{B})^{i-2}\frac{(B+1)^{N+1}-1}{B} \leq (B+1)^N\sum_{i=1}^{B}(\frac{B+1}{B})^{i-1} \\ &= (B+1)^N B((1+1/B)^B - 1) = mB((1+1/B)^B - 1).\end{aligned}$$

Since the initial buffer configuration stores mB packets, we have $T_{ADV} \leq mB(1+\frac{1}{B})^B$ and we conclude $c \geq 1/(1-(1+1/B)^{-B}+N^B/m)$ and for large B and N this expression can be arbitrarily close to $e/(e-1)$. □

Lemmas 3 and 4 establish the theorem. □

For $B=2$ we can show a stronger bound.

Theorem 3 *For $B=2$, no deterministic online algorithm can achieve a competitiveness smaller than $13/7$.*

Proof. We enhance the request sequence σ presented in the proof of Theorem 2. Let there be m additional queues for both *ADV* and *ALG*. At the beginning, there arrives one packet in each of the $2m$ queues. During the first m time steps, *ADV* serves those queues not served by *ALG* during this period. Then, σ is applied to those queues already served by *ADV*. Let T_{ADV} and T_{ALG} denote the throughputs of *ADV* and *ALG*, respectively, for σ, again, and let T'_{ADV} and T'_{ALG} denote the respective throughputs for the composed request sequence. Since $T'_{ADV} = T_{ADV} + 2m$ and $T'_{ALG} = T_{ALG} + m$, the throughput difference for any online algorithm is the same as for GR. Hence, we can apply Lemma 3 again and adopt the proof of Lemma 4. Let T_G and T'_G denote the respective throughputs for GR. Since $T'_G = T'_{ADV} - (m+1)B + S_{ADV}$ and $mB \leq T'_{ADV} \leq m(B(1+\frac{1}{B})^B + 2)$, we get for the competitive ratio that

$$c \geq T'_{ADV}/T'_G \geq 1/\left(1 - \frac{m(B+1)}{T_{ADV}} + \frac{N^B}{m}\right) \geq 1/(1-(B+1)(B(1+\frac{1}{B})^B+2)^{-1} + N^B/m),$$

taking values arbitrarily close to $1/(1-(B+1)(B(1+\frac{1}{B})^B+2)^{-1})$ for large N. Putting $B=2$ yields $c \geq 1/(1-3(2(1+\frac{1}{2})^2+2)^{-1}) = 1/(1-3(\frac{13}{2})^{-1}) = 1/(1-\frac{6}{13}) = \frac{13}{7}$. □

We next consider randomized algorithms. Azar and Richter [3] showed that if $B=1$, no randomized online algorithm can achieve a competitive ratio smaller than 1.4659. We prove the same lower bound for any buffer size B.

Theorem 4 *If ALG is a randomized online algorithm, its competitive ratio cannot be smaller than 1.4659, for any buffer size B.*

Proof. Let each queue be populated by B packets at the beginning. We call B subsequent time steps a round. At the end of each round, B packets arrive in the queue currently having the expected maximum load. Since the adversary *ADV* knows where the next packets will arrive, it serves this queue during the preceding B time steps. Hence, *ADV* can accept all packets whereas *ALG* has an expected loss of l_i packets in round i. First, we show that

$$\sum_{i=1}^{n} l_i \geq \sum_{i=1}^{n}\left(\frac{m-1}{m}\right)^i B. \tag{1}$$

Then, we will derive three lemmas from which we eventually conclude our claim.

We show (1) by induction on i. Let $i = 1$. After B time steps, *ALG* has a population of $mB - B = (m-1)B$ packets. Hence, the average queue length is $\frac{m-1}{m}B$. So, the expected maximum queue length is at least $\frac{m-1}{m}B$, yielding an expected loss $l_1 \geq \frac{m-1}{m}B$. Now assume $\sum_{i=1}^{j} l_i \geq \sum_{i=1}^{j}(\frac{m-1}{m})^i B$. Until the end of round $j+1$, *ALG* has transmitted $(j+1)B$ packets while its expected number of accepted packets is $jB - \sum_{i=1}^{j} l_i$. Hence, its expected buffer population is

$$mB - (j+1)B + jB - \sum_{i=1}^{j} l_i = (m-1)B - \sum_{i=1}^{j} l_i.$$

So, the expected average queue length is

$$\frac{1}{m}((m-1)B - \sum_{i=1}^{j} l_i) = \frac{m-1}{m}B - \frac{1}{m}\sum_{i=1}^{j} l_i.$$

We derive that the expected maximum queue length after round $j+1$ is at least $\frac{m-1}{m}B - \frac{1}{m}\sum_{i=1}^{j} l_i$, yielding an expected loss $l_{j+1} \geq \frac{m-1}{m}B - \frac{1}{m}\sum_{i=1}^{j} l_i$. Hence

$$\begin{aligned}\sum_{i=1}^{j+1} l_i &\geq \sum_{i=1}^{j} l_i + \frac{m-1}{m}B - \frac{1}{m}\sum_{i=1}^{j} l_i = \frac{m-1}{m}(B + \sum_{i=1}^{j} l_i) \\ &\geq \frac{m-1}{m}(B + \sum_{i=1}^{j}(\frac{m-1}{m})^i B) = \sum_{i=1}^{j+1}\left(\frac{m-1}{m}\right)^i B.\end{aligned}$$

If there are n rounds, then the throughputs of *ADV* and *ALG* are $T_{ADV} = (m+n)B$ and $E[T_{ALG}] = T_{ADV} - \sum_{i=1}^{n} l_i \leq (m+n)B - \sum_{i=1}^{n}(\frac{m-1}{m})^i B$ giving the ratio

$$r = \frac{T_{ADV}}{E[T_{ALG}]} \geq \frac{(m+n)B}{(m+n)B - \sum_{i=1}^{n}(\frac{m-1}{m})^i B} = \frac{m+n}{m+n - \sum_{i=1}^{n}(\frac{m-1}{m})^i}.$$

The number of rounds n is to be chosen such that r is maximized. Hence we want to maximize $\frac{1}{m+n}\sum_{i=1}^{n}(\frac{m-1}{m})^i$. In order to analyze this ratio we shall prove some lemmas.

Let $m \in \mathbf{N}$, $0 < q < 1$. For $n \in \mathbf{N}$, we define $a_n := \frac{1}{n+m}\sum_{i=1}^{n} q^i$. Since $a_n = \frac{1}{n+m}\frac{q-q^{n+1}}{1-q}$, we extend $(a_n)_n$ to $a_x = \frac{1}{x+m}\frac{q-q^{x+1}}{1-q}$ for all positive reals x.

Lemma 5 *If $a_{n+1} < a_n$, then $a_{n+2} < a_{n+1}$.*

Proof. Let $a_{n+1} < a_n$. Algebraic manipulations give $(m+n)\sum_{i=1}^{n+1} q^i < (m+n+1)\sum_{i=1}^{n} q^i$ and $(m+n)q^{n+1} < \sum_{i=1}^{n} q^i$. Using the latter inequality we obtain

$$(m+n+1)q^{n+2} < (m+n)q^{n+1} + q^{n+2} < \sum_{i=1}^{n} q^i + q^{n+2} < \sum_{i=1}^{n+1} q^i.$$

This implies

$$(m+n+1)\sum_{i=1}^{n+1} q^i + (m+n+1)q^{n+2} < (m+n+1)\sum_{i=1}^{n+1} q^i + \sum_{i=1}^{n+1} q^i = (m+n+2)\sum_{i=1}^{n+1} q^i.$$

We conclude

$$(m+n+1)\sum_{i=1}^{n+2} q^i < (m+n+2)\sum_{i=1}^{n+1} q^i$$

and finally, as desired,

$$\frac{1}{m+n+2}\sum_{i=1}^{n+2} q^i < \frac{1}{m+n+1}\sum_{i=1}^{n+1} q^i.$$

□

Corollary 1 *There exists a unique $n \in \mathbf{N}$ such that $a_{n-1} \le a_n$ and $a_n > a_{n+1}$. For this n, there holds $a_n = \max_{r\in\mathbf{N}} a_r$.*

Lemma 6 *There is a unique $n \in \mathbf{N}$ such that $q < \frac{a_n}{q^n} \le 1$.*

Proof. By Corollary 1, there exists a unique $n \in \mathbf{N}$ such that $a_{n-1} \le a_n$ and $a_n > a_{n+1}$. These two inequalities are equivalent to $\sum_{i=1}^{n-1} q^i \le (m+n-1)q^n$ and $\sum_{i=1}^{n} q^i > (m+n)q^{n+1}$. This is equivalent to $(m+n)q^{n+1} < \sum_{i=1}^{n} q^i \le (m+n)q^n$. Dividing by q^n and $m+n$, we obtain the inequality to be proven. □

Corollary 2 *$a_n = \max\limits_{r\in\mathbf{N}} a_r$ if and only if $q < \frac{a_n}{q^n} \le 1$.*

Lemma 7 *If $q = (m-1)/m$ and $a_{n(m)} = \max_{r\in\mathbf{N}} a_r$, then $\lim_{m\to\infty} n(m)/m = x$, where x is the unique positive solution of the equation $e^x = x+2$.*

Proof. We fix m and look for n such that $a_n/q^n = 1$. Then, by Corollary 2, $n(m) = \lfloor n \rfloor$. Define $\tilde{q}$ by $q\tilde{q} = 1$. We have $\tilde{q} - 1 = m/(m-1) - 1 = 1/(m-1)$ and $m = \tilde{q}/(\tilde{q}-1)$. The equation $\frac{a_n}{q^n} = 1$ is equivalent to $\sum_{i=1}^{n} q^i = (m+n)q^n$, which in turn is equivalent to $\sum_{i=1}^{n} \tilde{q}^{-i} = (m+n)\tilde{q}^{-n}$. We obtain $\sum_{i=0}^{n-1} \tilde{q}^i = m+n$, and hence $\tilde{q}^n - 1 = (m+n)/(m-1)$. Thus $\tilde{q}^n = (2m+n-1)/(m-1)$. Since $m = \tilde{q}/(\tilde{q}-1)$ we have

$$\tilde{q}^n = (2\frac{\tilde{q}}{\tilde{q}-1} + n - 1)/(\frac{\tilde{q}}{\tilde{q}-1} - 1) = 2\tilde{q} + (n-1)(\tilde{q}-1) = (\tilde{q}-1)n + (\tilde{q}+1).$$

Define α by $n = \alpha m$.

We conclude that the inequality $a_n/q^n = 1$ is equivalent to $\tilde{q}^n = (\tilde{q}-1)n + (\tilde{q}+1)$, which in turn is equivalent to

$$\left(\left(\frac{m}{m-1}\right)^m\right)^\alpha = \left(\frac{m}{m-1} - 1\right)\alpha m + \left(\frac{m}{m-1} + 1\right) = \frac{m}{m-1}\alpha + \frac{2m-1}{m-1}.$$

For $m \to \infty$, this equation takes the form $e^\alpha = \alpha + 2$. □

It remains to determine the competitive ratio r. We have

$$a_{n(m)} = \frac{\sum_{i=1}^{\lfloor \alpha_m m \rfloor} (\frac{m-1}{m})^i}{m + \lfloor \alpha_m m \rfloor} \ge \frac{m-1}{m}\frac{1 - (\frac{m-1}{m})^{m\alpha_m - 1}}{1 - \frac{m-1}{m}}\frac{1}{m + \lfloor \alpha_m m \rfloor} \underset{m\to\infty}{\longrightarrow} \frac{1-e^{-\alpha}}{1+\alpha} =: p.$$

For the ratio r, there holds

$$r \ge 1/(1-p) = 1/\left(1 - \frac{1-e^{-\alpha}}{1+\alpha}\right) = \frac{1+\alpha}{1+\alpha-1+e^{-\alpha}} = \frac{1+\alpha}{1+\alpha e^\alpha}e^\alpha.$$

Since $e^\alpha = \alpha + 2$, we get

$$r = \frac{(1+\alpha)(\alpha+2)}{1+\alpha(\alpha+2)} = \frac{(1+\alpha)(\alpha+2)}{\alpha^2+2\alpha+1} = \frac{(\alpha+1)(\alpha+2)}{(\alpha+1)^2} = \frac{\alpha+2}{\alpha+1} = 1 + \frac{1}{\alpha+1}.$$

Numerically solving equation $e^\alpha = \alpha + 2$ yields $\alpha \approx 1.1462$ and $r \approx 1.4659$. □

3 Upper bounds

We first present our new algorithm *SGR* and then study the influence of resource augmentation.

3.1 A New Semi-Greedy Algorithm

Algorithm Semi-Greedy (SGR): In each time step the algorithm executes the first rule that applies to the current buffer configuration.

1. If there is a queue buffering more than $\lfloor B/2 \rfloor$ packets, serve the queue currently having the maximum load.
2. If there is a queue the hitherto maximum load of which is less than B, serve amongst these queues the one currently having the maximum load.
3. Serve the queue currently having the maximum load.

Ties are broken by choosing the queue with the smallest index. The hitherto maximum load is reset to 0 for all queues whenever all queues are unpopulated in *SGR*'s configuration.

In the remainder of this subsection we analyze *SGR*. Let σ be any packet arrival sequence. We divide σ into several phases where phase P_1 starts at time $t = 0$. Phase P_i ends when *SGR* has completely cleared its buffer for the ith time, and phase P_{i+1} starts when the first packet arrives after the end of phase P_i. W.l.o.g. we can assume that at the beginning of each phase *ADV*'s buffer is empty as well because we can always postpone the start of the subsequent phase until this state is reached, thus only enlarging *ADV*'s throughput and hence the throughput ratio. So, the total throughput is given by the sum of the throughputs in the distinct phases if these were served separately. We derive that the ratio of the total throughputs cannot be worse than the throughput ratio in a single phase. Hence we can restrict ourselves to arrival sequences that terminate when *SGR* has completely cleared its buffer. Furthermore it suffices to consider sequences where at any time step, if there is packet loss at queue i, either *SGR* or *ADV* loses packets. If both lose packets, then we can reduce the number of packets arriving at i by the minimum loss of any of the two algorithms.

For the analysis of *SGR* we introduce a new potential function. Let $l_{it,A}$ and $l_{it,S}$ be the load of queue i at time t in the configurations of *ADV* and *SGR*, respectively. We define the potential of the queue as $\Phi_{it} = (l_{it,A} - l_{it,S})_+$ where $x_+ = (x + |x|)/2$. The total potential at time t is given by $\Phi_t = \sum_{i=1}^{m} \Phi_{it}$. Let T be the time when *SGR* has completely emptied its buffer. Since both *ADV* and *SGR* transmit one packet in each time step until T, the potential Φ_T describes their throughput difference. By $\Delta\Phi$ we denote the potential change $\Phi_t - \Phi_{t-1}$. We assume that during each time step, the arrival step precedes the transmission step.

Lemma 8 *The potential Φ does not increase during the arrival step, but Φ decreases by the number of packets lost by ADV.*

Proof. Let p packets arrive at queue i at time step t. If neither *ADV* nor *SGR* loses any packet, there holds $l_{it,A} = l_{i,t-1,A} + p$, $l_{it,S} = l_{i,t-1,S} + p$, where we measure the load at $t-1$ after the transmission step. Hence $\Phi_{it} = (l_{it,A} - l_{it,S})_+ = ((l_{i,t-1,A} + p) - (l_{i,t-1,S} + p))_+ = (l_{i,t-1,A} - l_{i,t-1,S})_+ = \Phi_{i,t-1}$ and the potential does not change due to the packet arrival at queue i. As mentioned above we can restrict ourselves to arrival sequences where either *ADV* or *SGR* loses packets whenever packet loss occurs. First we assume that *SGR* loses some packets. We derive $l_{i,t-1,A} < l_{i,t-1,S}$ and, hence, $\Phi_{i,t-1} = 0$. After the arrival of the p packets, there holds $l_{it,S} = B$ yielding $l_{it,A} \leq l_{it,S}$ and, hence, $\Phi_{it} = 0 = \Phi_{i,t-1}$. Since the single potentials do not change, nor does the total potential. Now we assume that *ADV* loses p_0 of the arriving p packets. Then $\Phi_{i,t-1} = (l_{i,t-1,A} - l_{i,t-1,S})_+ \geq p_0$. After the arrival of the p packets, there hold $l_{it,A} = B = l_{i,t-1,A} + (p - p_0)$ and $l_{it,S} = l_{i,t-1,S} + p$ yielding $(l_{it,A} - l_{it,S})_+ = (l_{i,t-1,A} + (p - p_0) - l_{i,t-1,S} - p)_+ = (l_{i,t-1,A} - l_{i,t-1,S} - p_0)_+ = (l_{i,t-1,A} - l_{i,t-1,S})_+ - p_0$ and, hence, $\Phi_{it} = \Phi_{i,t-1} - p_0$. □

Lemma 9 *During each transmission step the potential can change by at most 1.*

Proof. If *ADV* and *SGR* serve the same queue, the potential does not change at all. Let *ADV* serve queue i while *SGR* serves queue $j \neq i$. Let t^a and t be the moments after the arrival and the transmission steps, respectively. $\Phi_{it} = (l_{it,A} - l_{it,S})_+ = (l_{it^a,A} - 1 - l_{it^a,S})_+ = (l_{it^a,A} - l_{it^a,S} - 1)_+$ and $\Phi_{jt} = (l_{jt,A} - l_{jt,S})_+ = (l_{jt^a,A} - (l_{jt^a,S} - 1))_+ = (l_{jt^a,A} - l_{jt^a,S} + 1)_+$. Since $0 \leq (x+1)_+ - x_+ \leq 1$, we derive $-1 \leq \Phi_{it} - \Phi_{it^a} \leq 0$ and $0 \leq \Phi_{jt} - \Phi_{jt^a} \leq 1$. Hence there holds $-1 \leq \Phi_t - \Phi_{t^a} \leq 1$. □

Lemma 10 *During transmission step t the potential increases if and only if ADV serves a queue currently having potential 0 while SGR serves a queue having positive potential afterwards.*

Proof. Let the potential increase by 1. Let *ADV* serve queue i while *SGR* serves queue $j \neq i$. As shown in Lemma 9, $-1 \leq \Delta\Phi_{it} \leq 0$ and $0 \leq \Delta\Phi_{jt} \leq 1$. Hence the potential increases by 1 if and only if $\Delta\Phi_{it} = 0$ and $\Delta\Phi_{jt} = 1$. On the one hand, if $\Phi_{i,t-1} > 0$, there holds $\Phi_{it} = \Phi_{i,t-1} - 1$ and, hence, $\Delta\Phi_{it} = -1$. On the other hand, if $\Phi_{jt} = 0$, there holds $\Phi_{j,t-1} = 0$ and, hence, $\Delta\Phi_{jt} = 0$. So, the potential increases if and only if $\Phi_{i,t-1} = 0$ and $\Phi_{jt} > 0$. □

If the potential increases, the queue served by *SGR* is called the *causal queue* whereas the one served by *ADV* is termed *counter queue*. Whenever the potential increases, the situation of *ADV* worsens in such a way that *ADV* buffers more packets in queue j than *SGR* does. But, the same is true, mutatis mutandis, for *SGR* and queue i. The difference of the number of packets held by *ADV* and *SGR* in queue i increases whenever Φ increases and *ADV* serves i. Hence, Φ measures the number of packets that *SGR* has already lost or could lose if *ADV* replenishes the corresponding queues. These replenishments are necessary to generate a larger throughput for *ADV*.

If t is a time step where the potential does not increase during packet transmission, we call t an *additional time step* and denote their number by T_0. Hence, Φ increases $T - T_0$ times. Let T_{-1} be the number of time steps where the potential decreases during packet transmission. Obviously, $T_{-1} \leq T_0$. Furthermore, we derive from Lemma 8 that Φ decreases by L_0 during the arrival steps where L_0 is the total number of packets lost by *ADV*. We have $\Phi_T = T - T_0 - T_{-1} - L_0$. Let T_A and T_S denote the throughputs of *ADV* and *SGR*, respectively. From $T_A = T_S + \Phi_T$ and $T_S = T$, we derive

$$\frac{T_A}{T_S} = \frac{T + \Phi_T}{T} \leq \frac{T + T - T_0 - T_{-1} - L_0}{T} = 2 - \frac{T_0 + T_{-1} + L_0}{T}. \tag{2}$$

In the following, we shall present situations where the potential does not increase and hence derive that $\frac{T_0 + T_{-1} + L_0}{T} \geq \frac{1}{9}$, establishing the $(2 - \frac{1}{9})$-competitiveness of *SGR*.

We introduce some further notations. The queues are divided into B sets $Q^1, \ldots, Q^B$ where queue q members Q^k if and only if k is the maximum load of q in *SGR*'s configuration while processing σ. Furthermore, depending on *SGR*'s configuration at q, we call q a Q_2*-queue* if *SGR* currently buffers at least $\left\lfloor \frac{B}{2} \right\rfloor + 1$ packets in q. In case of *SGR*'s buffering at most $\left\lfloor \frac{B}{2} \right\rfloor$ packets in q, we call q a Q_{12}*-queue* if q has already had a total load of B packets, i.e. all buffers have been populated, otherwise we call it a Q_{11}*-queue*. By s_{11}, s_{12}, and s_2 we denote the number of time steps where the potential increases while *SGR* serves a Q_{11}, Q_{12}, and Q_2-queue, respectively. We partition s_{11} into s_{111} and s_{112} where we assign an s_{11} time step to s_{111} if the counter queue is a Q_{11}-queue in *SGR*'s configuration and to s_{112}, otherwise. Finally, let $b = \left\lfloor \frac{B}{2} \right\rfloor$, $L = \{1, \ldots, b\}$, $R = \{b+1, \ldots, B\}$ and $R^* = R \setminus \{B\}$.

Lemma 11 *There holds the inequality*

$$L_0 + T_0 \geq (s_2 - (B - b) \sum_{k \in R} Q^k)_+.$$

Proof. Let $k \in R$ and $j \in Q^k$ be a queue *SGR*'s serving of which has already increased s_2 at least $B - b$ times. Hence, at least B packets have arrived at j. For each further increase of s_2, a buffer in queue j must have been repopulated due to a further packet arrival. If *ADV* can accept the arriving packet, *ADV* must have served j, too. In these time steps, *ADV* serves a queue having positive potential. Hence, they are additional. If *ADV* cannot accept the arriving packet, *ADV* has a further packet loss. Hence s_2 can increase at most $B - b$ times at *SGR*'s serving j without additional time steps or packet loss. □

Lemma 12 *There holds the inequality*

$$T_0 \geq (s_{111} - b|Q^B|)_+ + s_{112} + s_{12} + (B - 2b)|Q^B| - T_{-1}.$$

Proof. We investigate s_{11} and s_{12} increases and focus on steps in which queue i is counter queue. We first study the case that i is a Q_{11}-queue and then consider the setting that i is a Q_{12}-queue.

We first observe that if i is a Q_{11}-queue, then by the definition of *SGR* the potential increase cannot be one assigned to s_{12}. Suppose that i is used n_i times as a counter queue for an s_{11} increase while i is a Q_{11} queue itself. We denote these time steps by $t_1 < \ldots < t_{n_i}$. First, we assume that $i \in Q^k, k < B$. Hence, *SGR* has no packet loss in i. Due to Lemma 10, *ADV* buffers at most as many packets in i as *SGR* before the transmission steps at $t_1, \ldots, t_{n_i}$. Since each packet is transmitted only once, there exist time steps $t'_1 < \ldots < t'_{n_i}$, with $t'_j > t_j, 1 \leq j \leq n_i$, where *SGR* serves i while *ADV* does not and *SGR* buffers more packets than *ADV* in i. These time steps are additional. Now, assume that $i \in Q^B$ and that *SGR* buffers B packets in i for the first time at time t. Since i cannot be a Q_{11} queue after t, we derive that $t_{n_i} < t$. At each $t_j, 1 \leq j \leq n_i$, *SGR* buffers at most b packets in i. We claim that there exist time steps $t'_1 < \ldots < t'_{n_i - b} < t$, with $t'_j > t_j, 1 \leq j \leq n_i - b$, where *SGR* serves queue i while *ADV* does not and *SGR* buffers more packets than *ADV* in queue i. These time steps are additional. To see the claim, take the time steps t_j in increasing order and match each one with the next unmatched time step $t'_j > t_j$ at which *SGR* serves i. Clearly, the unmatched t_j form a consecutive subsequence of $t_1, \ldots, t_{n_i}$ that includes t_{n_i}. Each unmatched time step corresponds to an increase of 1 in the buffer population difference between *SGR* and *ADV*. If more than b time steps were unmatched, then *SGR* would buffer more than b packets in i at time t_{n_i}, contradicting the fact that i is a Q_{12}-queue at that time. Hence, at most the b time steps $t_j, n_i - b < j \leq n_i$ are not matched. Combining the arguments of this paragraph, we derive that $T_0 \geq \left(s_{111} - b|Q^B|\right)_+$.

Now, we assume that at time t', queue i is a Q_{12}-queue and used as a counter queue for an s_{11} or s_{12} increase. Then, there holds $i \in Q^B$ and $t' \geq t$. By Lemma 10, in both *ADV*'s and *SGR*'s configurations, only the L-part is populated in queue i. If i is used as a counter queue during $n'_i > b$ increases of s_{11} or s_{12} at all of which it is a Q_{12}-queue, then, as above, we can prove that there are at least $n'_i - b$ time steps where *SGR* serves queue i while *ADV* does not and *SGR* buffers more packets than *ADV* in queue i. These time steps are additional. Moreover, these time steps are different from those identified in the last paragraph because they occur after t.

Next, we consider the first use of i being a Q_{12}-queue as a counter queue for an s_{11} or s_{12} increase. Consider the first $B - b$ time steps after t satisfying one of the following properties: (a) both *SGR* and *ADV* serve i; (b) only the algorithm currently having the larger buffer population in i serves i. These time steps exist and all occur before the first use of i being a Q_{12}-queue because *SGR* serves i at least $B - b$ times during that time window. The $B - b$ time steps are additional. However, they need not be distinct for the various queues i and i' and need not be distinct from the additional time steps identified so far. If a time step is counted twice, *ADV* serves i having a larger population than *SGR* in that queue, while *SGR* serves i' having a larger population in there. Note that the potential drops during packet transmission in this case. In order not to count one and the same time step twice when summing up over all queues, we diminish this amount of time steps by T_{-1}. Since $n'_i - b + B - b = n'_i + (B - 2b)$, combining the arguments of the three paragraphs, we obtain the lemma. □

While the two lemmas above cover the case that there are a lot of time steps where the potential increases, the following one considers the opposite situation.

Lemma 13 *There holds the inequality*

$$T_0 \geq T_{0,11} + T_{0,12} + T_{0,2}$$

where

$$\begin{aligned} T_{0,11} &= (\sum_{k\in L} k|Q^k| + \sum_{k\in R^*} b|Q^k| - s_{11})_+ \\ T_{0,12} &= (b|Q^B| - s_{12})_+ \\ T_{0,2} &= (\sum_{k\in R} (k-b)|Q^k| - s_2)_+. \end{aligned}$$

Proof. *SGR* must serve each buffer cell that is populated at least once. If $k \in L$ and $i \in Q^k$, in *SGR*'s configuration, the populated part only consists of the k first cells and i is always a Q_{11}-queue and, hence, can only cause s_{11} increases. If $k \in R^*$ and $i \in Q^k$, in *SGR*'s configuration, i is always a Q_{11}-queue when the j^{th} cell, $1 \leq j \leq b$, becomes unpopulated and, hence, can only cause s_{11} increases at these time steps. If $i \in Q^B$, in *SGR*'s configuration, i is always a Q_{12}-queue when the j^{th} cell, $1 \leq j \leq b$, becomes unpopulated after the buffer has once been populated by B packets and can only cause s_{12} increases afterwards. If $k \in R$ and $i \in Q^k$, in *SGR*'s configuration, the populated R-part consists of $k-b$ cells. There must be additional time steps if the potential increases less often than the number of populated buffer cells. □

Lemma 14 *If $B \geq 2$, there hold the following inequalities:*

$$\begin{aligned} T_0 + T_{-1} &\geq s_{12} \\ 2T_0 + T_{-1} &\geq s_{11} + (B-2b)|Q^B| \\ L_0 + 5T_0 + 2T_{-1} &\geq s_2 - (B-2b)\sum_{k\in R^*} |Q^k|. \end{aligned}$$

Proof. The first inequality is an immediate result of Lemma 12. Summing up the inequalities of Lemmas 12 and 13 yields

$$\begin{aligned} 2T_0 + T_{-1} &\geq (s_{111} - b|Q^B|)_+ + s_{112} + s_{12} + (B-2b)|Q^B| + (b|Q^B| - s_{12})_+ \\ &\geq s_{111} + s_{112} + (-b + (B-2b) + b)|Q^B| \\ &= s_{11} + (B-2b)|Q^B|, \end{aligned}$$

establishing the second inequality. By the summation of the inequalities of Lemmas 11, 12 and 13, we get

$$\begin{aligned} L_0 + 3T_0 + T_{-1} &\geq (s_2 - (B-b)\sum_{k\in R} Q^k)_+ + s_{112} + s_{12} \\ &\quad +(B-2b)|Q^B| + (b|Q^B| - s_{12})_+ \\ &\quad +(\sum_{k\in L} k|Q^k| + \sum_{k\in R^*} b|Q^k| - s_{11})_+ \\ &\geq s_2 - s_{111} - (B-2b)\sum_{k\in R^*} |Q^k|. \end{aligned}$$

The addition of the second inequality of the lemma yields $L_0 + 5T_0 + 2T_{-1} \geq s_2 - (B-2b)\sum_{k\in R^*} |Q^k|$. □

Theorem 5 *SGR achieves a competitive ratio of* $17/9$.

Proof. Let Q denote $(B-2b)\sum_{k\in R^*}|Q^k|$. If $i \in Q^k$ and $k \in R^*$, at least $b+1$ packets arrive at i. Hence $T \geq (b+1)Q$, and, thus, $\frac{Q}{T} \leq \frac{1}{b+1}$. The summation of the three inequalities in Lemma 14 yields $L_0 + 8T_0 + 4T_{-1} \geq s_{11} + s_{12} + s_2 - Q$. We derive

$$L_0 + 9T_0 + 4T_{-1} \geq s_{11} + s_{12} + s_2 + T_0 - Q = T - Q$$

and, hence, $L_0 + T_0 + T_{-1} \geq \frac{1}{9}L_0 + T_0 + \frac{4}{9}T_{-1} \geq \frac{T-Q}{9}$, yielding

$$(L_0 + T_0 + T_{-1})/(T-Q) \geq 1/9.$$

If B is even, then $Q = 0$ and $(L_0 + T_0 + T_{-1})/T \geq 1/9$ and by equation (2) the theorem follows. If B is odd, $\frac{L_0+T_0+T_{-1}}{T} = \frac{L_0+T_0+T_{-1}}{T-Q}(1-\frac{Q}{T}) \geq \frac{1}{9}(1-\frac{1}{b+1}) = \frac{1}{9}(1-\frac{2}{B+1})$. Let $\delta_B = \frac{2}{B+1}$. Then $\lim_{B\to\infty}\delta_B = 0$ and we derive by equation (2) that

$$T_A/T_S \leq 2 - (1-\delta_B)/9 = 17/9 + \delta_B/9 \to 17/9 \text{ as } B \to \infty.$$

□

We can strengthen our analysis and show that, for $B = 2$, *SGR* is optimal.

Theorem 6 *For* $B = 2$, *SGR achieves a competitive ratio of* $13/7$.

Proof. If we put $B = 2$ into the inequalities of Lemmas 11, 12 and 13, they take the following forms: $L_0 + T_0 \geq (s_2 - |Q^2|)_+$, $T_0 \geq (s_{111} - |Q^2|)_+ + s_{112} + s_{12} - T_{-1}$ and $T_0 \geq (|Q^1| - s_{11})_+ + (|Q^2| - s_{12})_+ + (|Q^2| - s_2)_+$ from where we deduce the corresponding inequalities of Lemma 14 for $B = 2$: $T_0 + T_{-1} \geq s_{12}$ and $2T_0 + T_{-1} \geq s_{11}$. Summing up the 3 inequalities in Lemmas 11, 12 and 13, we can strengthen the third inequality of Lemma 14: $L_0 + 3T_0 + T_{-1} \geq (s_2 - |Q^2|)_+ + s_{12} + (|Q^2| - s_{12})_+ \geq s_2 - |Q^2| + s_{12} + |Q^2| - s_{12} = s_2$. Since $5T_0$ is replaced by $3T_0$ in the left hand side, we get a competitive factor of $\frac{13}{7}$ instead of $\frac{17}{9}$ in Theorem 5. □

3.2 Resource Augmentation

We first study the case that an online algorithm is granted additional buffer. Then we consider different transmission rates.

3.2.1 Additional Buffer

Let *ADV* and *ALG* have buffers of size B and $B + A$ for each queue, respectively. We denote the ratio of the additional buffer A and the standard buffer B by c.

Theorem 7 *Every reasonable online algorithm ALG having additional buffer* $A = cB$ *per queue is* $(\frac{c+2}{c+1})$*-competitive.*

Proof. Let σ be any arrival sequence. As in the analysis of *SGR* we can restrict ourselves to arrival sequences that terminate when *ALG* has completely cleared its buffer. Furthermore, we assume that in the case of packet loss at a queue either *ADV* or *ALG* loses packets. If there is no packet loss at all or if only *ADV* loses packets, then the throughput of *ALG* is at least at large as that of *ADV* and our claim holds. Hence we may assume that *ALG* loses packets. We partition the queues into two disjoint sets Q^ℓ and Q^o where Q^ℓ comprises exactly those ones where *ALG* loses packets when serving σ. The difference of the total packet

losses corresponds to *ADV*'s buffer population at the time when *ALG*'s buffer reaches the state of emptiness. Let T_{ADV} and T_{ALG} denote the throughputs of *ADV* and *ALG*, respectively. By L_{ADV} and L_{ALG} we denote the losses of *ADV* and *ALG*, respectively. We derive

$$T_{ADV} - T_{ALG} = L_{ALG} - L_{ADV} \leq |Q^\ell|B.$$

Furthermore, the total throughput is given by the throughputs in Q^o and Q^ℓ, denoted by $T^o_{ADV}, T^\ell_{ADV}, T^o_{ALG}$ and T^ℓ_{ALG}, respectively. So $T_{ADV} = T^o_{ADV} + T^\ell_{ADV}$ and $T_{ALG} = T^o_{ALG} + T^\ell_{ALG}$. Since there occurred packet loss in the queues membering Q^ℓ, at least $A + B = (1 + c)B$ packets must have arrived there, all of which could be accepted by *ALG*. Hence there holds

$$T^\ell_{ALG} \geq (1 + c)|Q^\ell|B,$$

yielding

$$\begin{aligned}\frac{T_{ADV}}{T_{ALG}} &= \frac{T_{ALG} + L_{ALG} - L_{ADV}}{T_{ALG}} = 1 + \frac{L_{ALG} - L_{ADV}}{T_{ALG}} = 1 + \frac{L_{ALG} - L_{ADV}}{T^o_{ALG} + T^\ell_{ALG}} \\ &\leq 1 + \frac{L_{ALG} - L_{ADV}}{T^\ell_{ALG}} \leq 1 + \frac{|Q^\ell|B}{(1 + c)|Q^\ell|B} = 1 + \frac{1}{1 + c} = \frac{c + 2}{c + 1}.\end{aligned}$$

□

Theorem 8 *For any greedy algorithm GR, the upper bound of $\frac{c+2}{c+1}$ is tight for all B and all $A = cB$.*

Proof. For all values of A and B we construct an instance with a throughput ratio arbitrarily close to $\frac{c+2}{c+1}$. First, we assume that both *ADV* and GR have buffers of size $B + A$. From the previous section we know that we can construct a staircase sequence such that *ADV* has a throughput of about $2m(B + A)$ whereas GR has a throughput of only $m(B + A)$ packets. While the staircase is built up in GR, *ADV* always serves the corresponding queues such that they are empty in *ADV*'s configuration. This results in the fact that *ADV* can accept additional $B + A$ packets in these queues. Although –now– the buffer of *ADV* is smaller than the one of GR, *ADV* can build up a staircase of level $B + A$ in GR's configuration. The difference only consists in *ADV*'s not being able to accept $B + A$, but only B additional packets per queue, while GR cannot accept any of them. Hence, GR accepts $B + A$ packets per queue, whereas *ADV* accepts $B + A + B = 2B + A$ packets per queue. This results in the following throughput ratio:

$$\frac{T_{ADV}}{T_{GR}} = \frac{2B + A}{B + A} = \frac{2B + cB}{B + cB} = \frac{c + 2}{c + 1}.$$

□

3.2.2 Increased Transmission Rate

Now we assume that *ADV* and *ALG* have the same buffer sizes, but *ALG* can transmit k packets per time step while *ADV* is still able to transmit only one packet per time step.

Theorem 9 *Every reasonable online algorithm ALG is $(1 + \frac{1}{k})$-competitive if its transmission rate is the k-fold of the adversary's one.*

Proof. As usual, we restrict ourselves to sequences which terminate when *ALG* has completely emptied its buffer and assume that in the case of packet loss at a queue either *ADV* or *ALG* loses packets. Let $l_{it,ADV}$ and $l_{it,ALG}$ be the loads (i.e. the numbers of packets) in queue i at time t in the configurations of *ADV* and *ALG*, respectively. By $L_{it,ADV}$ and $L_{it,ALG}$ we denote the numbers of packets *ADV* and *ALG* lose at

queue i at time t, respectively. Let $T_{ADV}, L_{ADV}, T_{ALG}$ and L_{ALG} denote the throughputs and losses of *ADV* and *ALG*, respectively. For each queue i we define a potential function $\Phi_{it} = (l_{it,ALG} - l_{it,ADV})_+$ which is summed up to $\Phi_t = \sum_{i=1}^{m} \Phi_{it}$. At time t, *ALG* can lose packets in queue i only if $\Phi_{it} > 0$. Then $\Delta\Phi_{it} = -L_{it,ALG}$. During each transmission step, Φ_{it} can increase only if *ADV* serves a queue not served by *ALG*. But if so, *ALG* transmits k packets from other queues, hence $\frac{1}{k}\Delta T_{t,ALG} \geq \Delta\Phi_{it}$. So we derive

$$\Delta\Phi_t \leq \frac{1}{k}\Delta T_{t,ALG} - L_{t,ALG}.$$

Let τ denote the time step when *ALG*'s buffer becomes empty. Since $\Phi_0 = 0 = \Phi_\tau$, there holds $0 \leq \frac{T_{ALG}}{k} - L_{ALG}$, hence $L_{ALG} \leq \frac{T_{ALG}}{k}$ giving us the following throughput ratio:

$$\frac{T_{ADV}}{T_{GR}} = \frac{T_{ALG} + (L_{ALG} - L_{ADV})}{T_{ALG}} \leq \frac{T_{ALG} + L_{ALG}}{T_{ALG}} \leq 1 + 1/k.$$

□

Theorem 10 *If $B = 1$, every reasonable online algorithm ALG having a transmission rate being the k-fold of the adversary's one has a competitive ratio of at least $1 + 1/k$.*

Proof. Let there be m queues, each of them populated at the beginning. While *ALG* serves $\frac{k}{k+1}m$ queues, *ADV* serves the remaining $\frac{1}{k+1}m$ ones. In the latter ones, one packet arrives at time $\frac{1}{k+1}m$. This pattern is repeated on $(\frac{1}{k+1})^2 m$, $(\frac{1}{k+1})^3 m, \ldots$ queues not served by *ALG* yet until the block size reaches 1. We get $T_{ADV}/T_{ALG} = \frac{1}{m}\sum_{j=0}^{i}(\frac{1}{k+1})^j m$ and this expression tends to $1/(1 - \frac{1}{k+1}) = 1 + 1/k$ as i goes to infinity. □

4 An optimal offline algorithm

We present an optimal offline algorithm for our unit value throughput problem.

Algorithm Shortest Forward Overflow Distance (SFOD): If there cannot be any buffer overflow any more, serve the queues in an arbitrary order. Otherwise select in each transmission step t a queue where the next overflow would occur if none of the queues were served. More precisely, let $l_{i,t}$ denote the length of queue i at time t. Determine t_0 such that $l_{i,t} + \sum_{\tau=t+1}^{t_0-1} \sigma_{i,\tau} \leq B$ for all i and $l_{i,t} + \sum_{\tau=t+1}^{t_0} \sigma_{i,\tau} > B$ for some queue i. If t_0 exists, select queue i; otherwise serve any queue.

The algorithm can be implemented so that it runs in linear time.

Theorem 11 *SFOD is an optimal offline algorithm.*

Proof. We prove the following two statements, which imply the theorem.

1. For each arrival sequence σ there exists an optimal schedule satisfying the *SFOD* rule.
2. Each schedule satisfying the *SFOD* rule is optimal.

We first prove statement 1. Let σ be any arrival sequence and let S be an optimal schedule for σ. We show that we can convert S into an *SFOD*-schedule S' without decreasing the throughput. If S satisfies the *SFOD* property, then there is nothing to show. Now assume that there exists a time step t where S does not satisfy *SFOD*. Let S serve queue i at time t and let queue j be a queue where the next buffer overflow would occur after t. S' is a copy of S, but at time t, S' serves queue j instead of i. Let t_0 denote the time step where the next overflow in queue j will occur if it is not served any more. Due to the *SFOD* rule, the next overflow in

queue j would occur before the next overflow in queue i. If S serves queue j at time t' with $t < t' < t_0$, S' serves queue i at time t'. Then at time t' both S and S' are again in the same configuration and their throughputs are the same. If S does not serve queue j until t_0, the buffer will overflow, and S will lose one packet more than S' will. If there is an overflow at queue i later on, S' will lose one packet more than S will. Then, the configurations of S and S' are the same, again, and yield to the same throughputs. Hence, in either case, we get $T_{S'} \geq T_S$. We can repeatedly apply this procedure until we obtain a schedule S' satisfying the *SFOD* rule. The procedure terminates due to the finiteness of σ.

We next show statement 2. If we start with an optimal schedule S, we will get an optimal schedule S' satsifying the *SFOD* rule. Let S be an *SFOD*-schedule and let $\hat{S}$ be an optimal *SFOD*-schedule. If there is a packet loss in S at time t, $\hat{S}$ must lose the same number of packets because in both schedules queues have been served where the next overflow was nearest in the future. Hence there can only be a difference between S and $\hat{S}$ if there are several queues whose next overflow is at the same time. If packet loss is inevitable at t, S and $\hat{S}$ lose the same number of packets at t because they served the queues concerned as often as possible. □

5 Open Problems

In this paper we have studied the problem of maximizing the throughput of unit-value packets at a switch with m input buffers. We have developed improved upper and lower bounds on the competitive performance of online algorithms. In particular, we have devised a strategy, called *Semi-Greedy*, that achieves a competitiveness of $\frac{17}{9}$ and is the first deterministic algorithm that beat the trivial upper bound of 2. An important problem is to determine tight upper and lower bounds on the performance of deterministic algorithms. Similarly, a challenging task is to determine the best possible performance of randomized solutions. Up to now we know of no randomized algorithms whose competitiveness is below the deterministic lower bound. Finally, it is interesting to study scenarios where data packets have values and we wish to maximize the total value of transmitted packets. As mentioned in the introduction, Azar and Richter [3] gave a general technique that transforms any c-competitive algorithm for a single queue into a $2c$-competitive algorithm for multi-queue systems. As a result, they derived competitive algorithms for various settings. It is conceivable that improved solutions are possible by investigating the respective settings directly.

References

[1] W. Aiello, Y. Mansour, S. Rajagopolan and A. Rosén, Competitive queue policies for differentiated services. *Proc. INFOCOM*, 431–440, 2000.

[2] N. Andelman, Y. Mansour and A. Zhu. Competitive queueing policies in QoS switches. *Proc. 14th ACM-SIAM Symp. on Discrete Algorithms*, 761–770, 2003.

[3] Y. Azar and Y. Richter. Management of multi-queue switches in QoS Networks. *Proc. 35th ACM Symp. on Theory of Computing*, 82–89, 2003.

[4] A. Aziz, A. Prakash and V. Ramachandran. A new optimal scheduler for switch-memory-switch routers. *Proc. 15th Annual ACM Symp. on Parallelism in Algorithms and Architectures*, 343–352, 2003.

[5] A. Bar-Noy, A. Freund, S. Landa and J. Naor. Competitive on-line switching policies. *Proc. 13th ACM-SIAM Symp. on Discrete Algorithms*, 525–534, 2002.

[6] E.L. Hahne, A. Kesselman and Y. Mansour. Competitive buffer management for shared-memory switches. *Proc. 13th ACM Symp. on Parallel Algorithms and Architectures*, 53–58, 2001.

[7] A. Kesselman, Z. Lotker, Y. Mansour, B. Patt-Shamir, B. Schieber and M. Sviridenko. Buffer overflow management in QoS switches. *Proc. 31st ACM Symp. on Theory of Computing*, 520–529, 2001.

[8] A. Kesselman and Y. Mansour. Loss-bounded analysis for differentiated services. *Proc. 12th ACM-SIAM Symp. on Discrete Algorithms*, 591–600, 2001.

[9] A. Kesselman, Y. Mansour and R. van Stee. Improved competitive guarantees for QoS buffering *Proc. 11th European Symp. on Algorithms*, LNCS Vol. 2832, 361–372, 2003.

[10] A. Kesselman and A. Rosén. Scheduling policies for CIOQ switches. *Proc. 15th Annual ACM Symp. on Parallelism in Algorithms and Architectures*, 353–361, 2003.

[11] H. Koga. Balanced scheduling towards loss-free packet queueing and delay fairness. *Proc. 12th Annual International Symp. on Algorithms and Computation*, LNCS Vol. 2223, 61–73, 2001.

[12] D.D. Sleator and R.E. Tarjan. Amortized efficiency of list update and paging rules. *Comm. of the ACM*, 28:202-208, 1985.