Distributed Online and Stochastic Queuing on a Multiple Access Channel*

Marcin Bienkowski Tomasz Jurdzinski Miroslaw Korzeniowski
Dariusz R. Kowalski

Abstract

We consider the problems of online and stochastic packet queuing in a distributed system of n nodes with queues, where the communication between the nodes is done via a multiple access channel. In the online setting, in each round, an arbitrary number of packets can be injected to nodes' queues. Two measures of performance are considered: the total number of packets in all queues, called the total load, and the maximum queue size, called the maximum load. We develop a deterministic distributed algorithm that is asymptotically optimal with respect to both complexity measures, in a competitive way. More precisely, the total load of our algorithm is bigger than the total load of any other algorithm, including centralized online solutions, by only an additive term of $O(n^2)$, while the maximum queue size of our algorithm is at most n times bigger than the maximum queue size of any other algorithm, with an extra additive $O(n)$. The optimality for both measures is justified by proving the corresponding lower bounds, which also separates nearly exponentially distributed solutions from the centralized ones. Next, we show that our algorithm is also stochastically stable for *any* expected injection rate smaller or equal to 1. This is the first solution to the stochastic queuing problem on a multiple access channel that achieves such stability for the (highest possible) rate equal to 1.

Keywords: distributed algorithms, online algorithms, multiple access channel, contention resolution, shared channel, stochastic queuing, stability

1 Introduction

Multiple Access Channel is one of the fundamental models for distributed communication. It has been widely studied and used in the context of theoretical analysis of Ethernet and wireless protocols, contention resolution in systems with buses, and in other emerging technologies. Roughly speaking, a multiple access channel models environments in which distributed nodes/resources compete for access to the shared communication and distribution channel, and in case of contention, no contender wins the access.

Distributed queuing on a multiple access channel is one of the most essential problems, widely studied by both theoreticians (cf. [12]) and practitioners (cf. [8]). In this problem, packets arrive continuously at nodes, and the goal is to maintain bounded queues and latency, whenever possible. The main two lines of research include design and analysis of protocols in two scenarios: for restricted adversarial injections and for stochastic ones. The best up-to-date results guarantee bounded queues only for *bounded* packet burstiness, in case of the former setting, and only for stochastic injection rates *strictly smaller* than 1 in the latter one. This work aims to resolve the remaining cases of heavy traffic, by pursuing a distributed online approach (cf. [6, 1]), and later adapting it also to the stochastic scenario (cf. [17, 12, 20]. We believe that the newly developed and analyzed distributed scheduling techniques could substantially improve flow stability in heavy

*A preliminary version of this paper appeared in the proceedings of the 26th International Symposium on Distributed Computing (DISC 2012). This work was supported by Polish National Science Centre grants DEC-2012/07/B/ST6/01534, DEC-2013/09/B/ST6/01538 and 2016/22/E/ST6/00499, and by the Engineering and Physical Sciences Research Council [grant number EP/G023018/1]. Authors' addresses: M. Bienkowski and T. Jurdzinski, Institute of of Computer Science, University of Wrocław, Poland; M. Korzeniowski, (Current address) MicroscopeIT, Wrocław, Poland; D. R. Kowalski, Department of Computer Science, University of Liverpool, UK.

traffic systems, such as data and video streaming under WLAN (Wireless Local Access Network) standard 802.11aa.

The model We follow the classical model of multiple access channel, cf. surveys by Gallager [17] and Chlebus [12]. That is, we consider the scenario where n nodes with pairwise disjoint identifiers (IDs) in $\{1, 2, \ldots, n\}$ broadcast packets and communicate through a multiple access channel (MAC). Each node has a buffer, also called a *queue*, of potentially infinite capacity. We assume slotted time, where time points are numbered from 0. At time 0, the adversary may inject an arbitrary number of packets, each placed at an arbitrary node. Then, for $t = 1, 2, 3, \ldots$, the following happens:

- In round t, defined as an interval between time $t-1$ and time t, any node may transmit a message containing at most one packet. A transmission is successful if exactly one node transmits in the round; in such case, the transmitted packet is removed from the queue of the transmitting node.
- Then, at time t, the adversary injects an arbitrary number of packets and place them in arbitrary queues.

Sometimes, to avoid ambiguity, we will use notion of t^- and t^+ to denote the time t right before and right after the adversary injects the packets.

We assume Ethernet-like capabilities of MAC, i.e., each node can simultaneously listen and transmit, and thus knows whether the transmission was successful or not. However, our positive results hold also in the model, where a station sending a message cannot listen at the same time. We assume that nodes can communicate only through MAC, but they are allowed to append control bits to the sent packets. We do not impose any restriction on the number of such bits, however our algorithm appends only $O(\log n)$ additional bits of information to a transmitted packet. Note that control bits are inevitable in order to achieve competitiveness even for restricted adversaries and $n \geq 3$ [13].

We consider two models of analysis of online queuing on a multiple access channel: competitive and stochastic. The former approach is new, in the sense that only injections of bounded burstiness have been considered so far, without comparison to the optimal solution.

Competitive ratio For any algorithm ALG and a packet injection pattern $\mathcal{I}$, let $Q_{\mathrm{ALG}}(\mathcal{I}, t, i)$ denote the length of the queue (the number of pending packets) at node i at time t^+. Let $L_{\mathrm{ALG}}(\mathcal{I}, t) = \sum_{i=1}^{n} Q_{\mathrm{ALG}}(\mathcal{I}, t, i)$ be the *total load* of ALG at time t under injection pattern $\mathcal{I}$. Finally, let $M_{\mathrm{ALG}}(\mathcal{I}, t) = \max_i Q_{\mathrm{ALG}}(\mathcal{I}, t, i)$ be the *maximum load* under injection pattern $\mathcal{I}$.

We call an online distributed algorithm (R, A)-competitive for minimizing the total load if for any adversarial pattern of packet injections $\mathcal{I}$ and any time t it holds that $L_{\mathrm{ALG}}(\mathcal{I}, t) \leq R \cdot L_{\mathrm{OPT}}(\mathcal{I}, t) + A$ where OPT is the optimal *offline centralized* solution for injection pattern $\mathcal{I}$ until time t. For randomized algorithms, in the definition above we replace $L_{\mathrm{ALG}}(\mathcal{I}, t)$ by $\mathbf{E}[L_{\mathrm{ALG}}(\mathcal{I}, t)]$, where the expectation is taken over all random choices of the algorithm up to the time t. That is, we consider oblivious adversaries [9] that cannot observe random choices made by an online algorithm and use this knowledge for generating an input sequence. We define competitiveness for minimizing the maximum load analogously, by replacing L_{ALG} and L_{OPT} by M_{ALG} and M_{OPT}, respectively, in the definitions above.

We emphasize that the relation between ALG and OPT has to hold for and *any* injection pattern $\mathcal{I}$ at *any* time t. Unlike in the traditional approach of competitive analysis [9], we explicitly give the additive term in the competitive ratio.

Stochastic queuing In the stochastic queueing, an adversary dictating the injection patterns is replaced by a stochastic process. Precisely speaking, such process is defined by a sequence of numbers $p_1, \ldots, p_n \in (0, 1)$. At each time $t \geq 0$, for each queue independently, one packet is injected into the queue i with probability p_i and no packet is injected to this queue with probability $1 - p_i$. The amount $\sum_{i=1}^{n} p_i$ is called *injection rate*. This way of modelling stochastic arrivals is popular in so called "queuing models" [20, 12], in which

the number of queues is fixed and for each packet a specific queue must be selected; this is the case of our model.[1]

The goal in such a scenario — for possibly largest injection rate — is to construct a distributed algorithm that achieves *stability*. Namely, we require that the algorithm reaches the state with empty queues infinitely many times with probability 1, regardless of the initial distribution of packets. We also introduce a similar notion of *strong stability*, meaning that from any initial distribution of packets the state of empty queues is reached in finitely many steps in expectation. Note that if the random execution defines a Markov Chain, then the stability is guaranteed once the Markov Chain is irreducible and recurrent, while the strong stability holds if the Markov Chain is irreducible and positive recurrent, which corresponds to ergodicity of Markov Chains, cf. [18].

1.1 Previous and related work

To the best of our knowledge, this is the first work studying online distributed queuing problem for unrestricted packet injection patterns. We analyze the competitive ratio of minimizing two important complexity measures: total load and maximum load. In what follows, we describe a related work including online queuing in the centralized model and queuing under restricted adversarial injection patterns. Next we provide a summary of stability results of protocols, so far obtained only for injection rates smaller than 1, cf. [20].

Online queuing in the centralized model The optimization problems considered in this paper were also analyzed in the setting where *central coordination* is assumed and all nodes have *global knowledge* about all injected packets. Clearly, minimizing the total load is no longer a challenge in such setting as any work-conserving algorithm (i.e., one that transmits packets from non-empty queue whenever possible) is optimal with respect to the total load minimization. However, minimizing the length of the maximal queue is non-trivial and known in the literature under the name of *balanced scheduling*. In particular, Fleischer and Koga [16], and independently Bar-Noy et al. [7], proved that any algorithm serving always a longest nonempty queue achieves the competitive ratio of $O(\log n)$. This ratio is asymptotically optimal even if we allow randomized solutions [16, 7]. Fleischer and Koga [16] proved additionally that the popular round-robin algorithm is $\Omega(m)$-competitive, where m is the number of injected packets. Note that it means that the round-robin algorithm is non-competitive in case of unbounded number of injected packets. The results for the centralized model and the distributed one, obtained in this work, is given in Table 1. In particular, one can see that the lack of central coordination significantly affects the performance of the whole system.

Online queuing in the distributed setting under restricted adversaries Inspired by adversarial queuing problems in store-and-forward packet networks [4, 10], several papers analyzed *distributed* queuing on a multiple access channel under *restricted* adversarial injection patterns. Previous works by Chlebus et al. [13, 14] and Anantharamu et al. [2] considered adversaries that were (ρ, b)-restricted, also called (ρ, b)-leaky-bucket, for $\rho \leq 1$ and $b \geq 1$. Such an adversary may only inject $\rho \cdot |I| + b$ packets into all queues in any time interval I. In [3] the authors considered a variant of this model in which each queue has its own restriction on arrival rate. Restricted adversaries were also used for modelling jamming on a multiple access channel [21].

We emphasize that the unrestricted adversary considered in this work may not only generate all injection patterns allowed for the restricted case, but also patterns with periods of "burstiness" growing arbitrarily large that were not allowed by the restricted adversary. Moreover, the previous results for the restricted adversary provided only time-global bounds on queue sizes, expressed as functions of parameters n, ρ, b, while the competitive analysis provided in this work compares the created solution to the optimal algorithm at any time. That is, for each time point t, the online solution is compared with the optimal offline centralized solution for injection pattern until t. Although algorithms designed and analyzed under the restricted adversarial

[1] Slightly different modelling, by Poisson arrivals, is used in so called "queue-free models", in which each packet is a separate queue and therefore it would be meaningless to decide to which "queue" it arrives, cf., a thorough discussion about models and results in surveys [12, 18].

Table 1: The bounds on the competitiveness in the centralized and distributed settings, for the two complexity measures: minimization of total load and maximum load.

	centralized	distributed
total load	OPT (straightforward)	$\text{OPT} + \Theta(n^2)$ (this paper)
maximum load	$\Theta(\log n) \cdot \text{OPT}$ [a]	$n \cdot \text{OPT} + O(n)$ (this paper)

injection patterns may not imply similar results in the more general competitive model considered in this work, some lower bounds can be adopted. In particular, Chlebus et al. [13] proved that, even in the $(1, 1)$-restricted setting, no algorithm achieves bounded latency. This implies that no algorithm is competitive with respect to the latency measure, and motivates our focus on the total and maximum load measures instead.

The scenario considered in this paper could also be seen as an online extension of the well-studied *contention resolution problem*. There, every node has a single packet and the goal is to transmit them all successfully on a shared channel. The focus is on minimizing packet latency as the queue loads are explicitly bounded by the problem definition. However, it is known that the packet latency is connected to the queues' lengths in the dynamic extension of this problem [14]. For recent developments, see [5, 15] and the references therein.

Stochastic queuing There is a rich history of research on *stochastic* queuing on a multiple access channels, i.e., when packets are injected subject to statistical constraints. See the surveys by Gallager [17] and Chlebus [12] for an overview of early research. In particular, Håstad et al. [20] proved that, for any fixed injection rate smaller than 1, polynomial backoff protocols are stable and exponential backoff ones are not stable. To the best of our knowledge, all the previous results concerning stability were proved for expected fixed injection rates *strictly smaller* than 1. Ours is the first deterministic distributed online algorithm achieving stability also for the (highest possible) injection rate 1.

1.2 Our Results

We develop a deterministic distributed online algorithm ScanTrim, which is $(n, 5n)$-competitive for the maximum load measure and $(1, n^2 + 4n)$-competitive for the total load measure. That is, the total load of our algorithm is, in each round, larger by an additive term of $O(n^2)$ than the total load of the best offline algorithm (taken for the same adversarial injection pattern). Moreover, the maximal queue size of our algorithm is, in each round, at most n times larger than the maximal queue size of the best offline algorithm, plus an additive term $O(n)$.

We show that both ratios are asymptotically optimal in the following sense. First, for any (R, A)-competitive algorithm minimizing maximum load, it holds that $R \geq n$ and $A = \Omega(n)$. Second, for any deterministic (R, A)-competitive algorithm minimizing total load, it holds that $R \geq 1$ and $A = \Omega(n^2)$. In both cases, bounds on R hold even for randomized algorithms.

See Table 1 for a summary of results concerning competitiveness of distributed online queuing on a multiple-access channel versus centralized online queuing. Although one can argue that such big bounds on the optimal values of competitiveness parameters diminish importance of our model, observe that these bounds do not depend on an execution length (number of rounds) of a protocol (which might be arbitrarily large).

Furthermore, we demonstrate the efficiency of our algorithm ScanTrim for the stochastic queuing problem, showing that it is stable for any injection rate not larger than 1. (Note that the injection rate 1 is the highest possible to obtain stability defined this way.) All previous solutions to this variant of the problem guaranteed such property only for the injection rate *strictly smaller* than 1. For this case, we show even a stronger property called *strong stability*: the expected number of rounds needed to reach the state with empty queues is finite.

1.3 Distributed online solution: challenges and ideas

Our main online algorithm ScanTrim is designed to overcome two fundamental challenges imposed by the shared channel: (i) delay in updating information (there is at most one node transmitting successfully at a time, hence most of the nodes transmitted $\Omega(n)$ rounds ago, and therefore other nodes may not know their current state) and (ii) *wasted rounds* (defined as rounds without successful packet transmission) that may occur during information gathering.

To demonstrate these issues, consider the behavior of already studied protocols. Probably the simplest one is the round-robin protocol, in which nodes transmit (and gather information) periodically according to a fixed order. It generates an unbounded number of wasted rounds when the adversary injects all packets to a single node, one packet per round. To reduce the number of wasted rounds, one could modify this protocol to empty the whole queue while visiting a node. However, an unbounded number of wasted rounds are then generated in a slightly more sophisticated scenario where the adversary injects one packet per round to a fixed node i until this node starts to be processed; then packets are injected to the node preceding i (again, one packet per round), and so on.

Another idea would be to use a queue to amortize the generated wasted rounds by checking the queues of potentially empty nodes: such an idea was introduced by Chlebus et al. [13], who proposed algorithm Move-Big-To-Front. In this algorithm, the round-robin procedure is applied until a node with a queue larger than n is found; in such case, the queue is moved to the beginning of the round-robin list and emptied in consecutive rounds down to the level of exactly n pending packets. Then, the round-robin sub-routine is applied again, and the whole process is repeated in a loop. Observe however that the adversary can first fill each node to the level of at least $n/2$ packets, by injecting one packet per round on average (see also the proof of the lower bound in [13]), and then — by injecting packets always to the last node on the list — create queues of size $\Omega(n^2)$. For such injection pattern, the optimum solution has always at most one packet in its queues.

The above examples are token-based protocols. Bianchi [8] argued that randomized backoff protocols are not stable under highly saturated injection patterns. In general, as we also demonstrate in the proof of one of our lower bounds, using non-coordinated transmission pattern may cause even more wasted rounds compared to the best offline solution, as collisions may occur due to simultaneous transmissions.

Our solution introduces a specific potential function that efficiently trades a delay in obtaining information about queue sizes for the (wasted) silent rounds. More precisely, the algorithm runs in two modes: *scanning* and *trimming*. The main goal of the former is to update the information, while in the latter the algorithm transmits packets in order to compete with the optimal solution. Based on the potential function, we define the order of scanned and trimmed nodes and the conditions for switching between the two modes.

The result in the stochastic injection setting is obtained by proving that an underlying Markov chain is (positive) recurrent. We show this in two steps. First, we define and analyze an Markov chain corresponding to the behavior of an offline solution. In particular, we show that prove that properties of this Markov chain imply stability and strong stability of the optimal offline protocol in the stochastic injection setting. Next, by applying the competitiveness result from the worst-case online analysis, we argue that the stochastic process corresponding to the execution of our online algorithm ScanTrim satisfies stability and strong stability.

2 Competitive Algorithm Scan-Trim

In this section, we construct a protocol ScanTrim and analyze its competitive ratio. We start with a high-level description and intuitions. Then, we provide the pseudo-code and the analysis.

The number of nodes n and the ID of a node are the only input parameters for the algorithm executed at the node. The protocol is *collision-avoiding*: it schedules transmissions in such a way that collisions never occur. To this end, it builds on a token-passing paradigm, in which a unique node with the "token-holder" status transmits a message. Recall that in the considered setting, a message contains at most one packet and a number of additional bits of information. In our protocol, the transmitting node attaches only the current size (i.e., the number of packets) of its queue computed right before the transmission (i.e., including

the transmitted packet).

We assume that if in a round the token holder has no packet to transmit, it still transmits a message, but it contains no packet, only the number zero representing its empty queue. Such a round we call *void*. This is for notational simplicity only — the same effect could be achieved by not transmitting anything in such rounds.

Our algorithm works with any local queuing policy (such as FIFO, LIFO, SIS, etc.) as it does not influence the considered measures of performance. In practice, FIFO queue could be seen as the most "fair" queuing policy.

Global state There is a certain number of variables stored by the algorithm at each node. In particular, each node keeps information which node holds the token, the current mode of operation (scanning or trimming, to be specified later), and the ordered list of all the nodes together with their queue sizes (taken at the time of their last transmission). While the exact description of these variables is given later, we emphasize that the values of these variables are the same for all nodes. This is achieved by (i) initializing all these variables to the same values, (ii) ensuring that their evolution is deterministic and depends solely on their current value and the information transmitted in a given round. Note that the information heard by all nodes is the same by the definition of the channel.

Hence, we call these variables *global*, keeping in mind that, in fact, they are stored locally, but their coherence is easily ensured.We also emphasize that except for these global variables and its own packet queue, no node holds any other information.

The most important global variable is a list $\mathcal{L}$ of nodes. It contains IDs of all the nodes stored in a certain order; the positions of $\mathcal{L}$ are numbered from 1 to n. Whenever we write "node i", we mean the node with the i-th position on list $\mathcal{L}$. Additionally, $\mathcal{L}$ stores three pieces of information for each node i:

- Key k_i, equal to the queue size of node i *right after* its last transmission. If node i has not yet transmitted, the key is set to zero.
- Queue non-emptiness indicator p_i equal to 1 if queue of node i was non-empty when it transmitted its last message (i.e., the round was non-void and an actual packet was sent) and 0 otherwise.
- Threshold value φ_i for node i equal to a non-negative integer, whose value will be determined later (and modified only at some particular rounds of the protocol).

Note that the amount $k_i + p_i$ is the number of packets in the queue of i right before the transmission (and the number actually transmitted in a message of node i), while k_i is the number of packets in this queue right after it. Apart from $\mathcal{L}$, there are two global variables, described in detail in the definition of the algorithm:

- *token*, a number i from $[1, n]$ denoting that the current token holder is the i-th node from $\mathcal{L}$;
- *mode*, which can be either *scanning* or *trimming*.

An intuition behind modes is as follows. In the scanning mode, a token is passed along the list $\mathcal{L}$ of all nodes. A node that has the token transmits one of its packets (if it has any) along with the current size of its packet queue and the token is adopted by the next node from $\mathcal{L}$. If the total number of the learned queue sizes is sufficiently large, all nodes switch to the trimming mode. Otherwise, after reaching the end of the list $\mathcal{L}$, the scanning mode continues from the first node of $\mathcal{L}$. In both cases, all nodes recompute the list $\mathcal{L}$ and threshold values φ_i based on the learned queue sizes. In the trimming mode, the token is passed also along the list $\mathcal{L}$, but only to nodes whose queue sizes are *known* to be larger than the fixed thresholds. In this mode, however, a token may be held by a node for multiple consecutive rounds until the number of packets in its queue reaches the threshold. All nodes switch back to the scanning mode once the total number of known packets becomes equal to the sum of all thresholds.

The main problem faced by the algorithm is the information delay: the keys stored in $\mathcal{L}$ are usually outdated as the nodes do not have the information about the recent changes to the queues made by packet

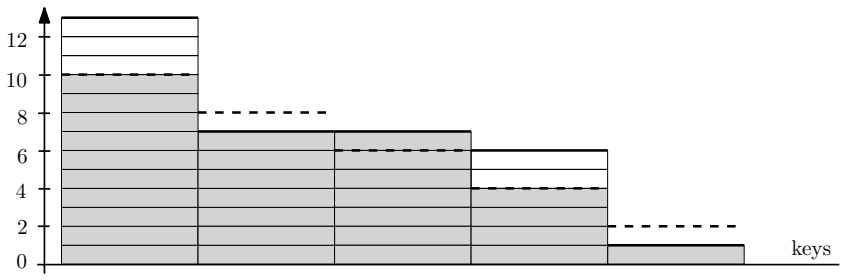

Figure 1: Example values of keys $k_1 \geq k_2 \geq \ldots \geq k_5$ and computed thresholds. Non-overhead packets are shaded. The values of $S_i - S_{i-1}$ (assuming $S_0 = 0$) are shown as dashed lines.

injections. Instead, $\mathcal{L}$ contains information about the queue sizes of a specific node from the round this node last transmitted. A great advantage of the global variables (representing only a partial knowledge) is that they allow for more coordinated approach, which is easier to analyze.

Potentials and their properties For any n non-negative integers $x_1, x_2, \ldots, x_n$ sorted in non-increasing order, we define a potential function $\pi : \{1, \ldots, n\} \to \mathbb{N} \cup \{0\}$. Function π is defined iteratively, from $i = 1$ up to $i = n$, as

$$\pi(i) = \min \left\{ x_i,\ S_i - \textstyle\sum_{j=1}^{i-1} \pi(j) \right\} \ ,$$

where $S_i = \sum_{j=1}^{i} 2(n+1-j) = (2n+1-i) \cdot i$. In particular, $\pi(1) = \min\{x_1, 2n\}$. We also define the total potential as $\overline{\pi} = \sum_{i=1}^{n} \pi(i)$.

An alternative, equivalent description of potentials computations is as follows. We keep a budget that is initially equal to zero. We iterate over all values of i from 1 to n. When considering index i, we first increase the budget by $S_i - S_{i-1} = 2(n+1-i)$. Then, we set $\pi(i) \leq x_i$ to be as large as possible, but the amount $\pi(i)$ has to be covered from the current budget.

The following facts are easily derived from the definition.

Fact 1. *For the potential function π of any values, it holds that*

1. $0 \leq \pi(i) \leq 2n$ *for any* $i \in [1, n]$,
2. $\pi(i) \geq \pi(i+1)$ *for any* $i \in [1, n-1]$,
3. $\sum_{j=1}^{i} \pi(j) \leq S_i$ *for any* $i \in [1, n]$.

Fact 2. $S_b - S_a = (b-a) \cdot (2n+1-a-b)$ *for any* $a, b \in [1, n]$.

Thresholds The algorithm uses the potential function to compute thresholds. (Recall that thresholds are recomputed, by all nodes simultaneously, only upon changing the mode from scanning to trimming and at the end of a scanning cycle.) At these points of time, $\mathcal{L}$ is sorted in the non-increasing order of keys. Ties are broken according to IDs, which ensures that the ordering is the same at all the nodes. Note that $k_1, k_2, \ldots, k_n$, denoting the values of keys, are now a non-increasing sequence. The potential π is computed on the values of keys k_i and is stored in the thresholds φ_i, i.e., we simply set $\varphi_i := \pi(i)$ for all $i \in [1, n]$. At any time, the packets at node i above the threshold φ_i are called *overhead packets*, i.e., node i with ℓ packets has $\max\{\ell - \varphi_i, 0\}$ overhead packets and $\min\{\varphi_i, \ell\}$ non-overhead packets. An example is presented in Figure 1.

Algorithm 1: Update of global variables at time t

```
case mode = scanning do
    if ∑_{i=1}^{token} (k_i + p_i − φ_i) ≤ token and token < n then
        token ← token + 1
    else
        sort L in a non-increasing order of keys
        φ ← the potential function of keys
        if there exists i such that k_i > φ_i then
            token ← min{i | k_i > φ_i}
            mode ← trimming
        else
            token ← 1
        end
    end
end
case mode = trimming do
    if ∑_{i=1}^{n} (k_i − φ_i) > 0 then
        if k_token ≤ φ_token then
            token ← min{i > token | k_i > φ_i}
        end
    else
        token ← 1
        mode ← scanning
    end
end
```

Algorithm definition At time 0, the algorithm initializes its global variables. Namely, $\mathcal{L}$ is populated with all nodes sorted by their IDs. Thresholds and keys for all nodes are set to zero, *token* is set to 1, and *mode* is set to *scanning*. Some packets may be already injected at time 0 by the adversary. Then, for $t = 1, 2, 3, \ldots$, the following happens (cf. Section 1 with the description of the model).

- In round t, the node whose position in $\mathcal{L}$ is equal to *token* transmits. It sends a message containing a packet from its queue (if it has any) along with the size of its queue (computed before removing the transmitted packet from the queue). All nodes, including the transmitting one receive this message.
- At time t, these three actions are performed at the nodes:
    1. All nodes update variables k_{token} and p_{token} on the basis of the message they heard in round t. Recall that $k_{token} + p_{token}$ is the number of packets in the queue of node *token* (and hence equal to the number transmitted in the message) and if this number was non-zero, then $p_{token} = 1$.
    2. The adversary injects an arbitrary number of packets; they are appended to particular queues.
    3. Each node executes Algorithm 1. Depending on the mode, all nodes execute the instructions from one of the two corresponding cases.

By this description, it is straightforward that all nodes are capable of tracking the values of global variables.

A remark on message size In the algorithm ScanTrim, a node holding a token attaches the current number of packets in its queue to the transmitted packet. The number of attached bits can be further optimized by restricting the attached number to the minimum from the current queue size and $2n + 1$. This modification may cause heavily overloaded nodes to be considered in different order than the one defined by their queue sizes. It can however easily be checked that the basic properties guaranteed by the execution

remain unchanged, as they are defined based on threshold values of at most $2n$, and consequently the whole analysis based on these properties still holds. This allows the algorithm to restrict the number of additional bits attached to each message to $O(\log n)$.

2.1 Framework of the Analysis

The goal of this and the remaining subsections is to show that the algorithm ScanTrim is optimal with respect to both complexity measures (maximum load and total load).

Consider an execution of algorithm ScanTrim. Its rounds can be grouped into *scanning* and *trimming* cycles in the following way. A *trimming cycle* is just a contiguous sequence of rounds in which ScanTrim is in the trimming mode. On the other hand, the contiguous sequence of rounds in which ScanTrim is in the scanning mode consists of one or more consecutive *scanning cycles*. Precisely, in the first round of the scanning cycle, the first node from $\mathcal{L}$ has the token, and the scanning cycle ends when the outer *else* branch is executed, i.e., when $\sum_{i=1}^{token}(k_i + p_i - \varphi_i) > token$ or $token = n$. If the former condition occurs, then we call such scanning cycle *balanced*. Intuitively, a scanning cycle is balanced if during this cycle we find sufficiently many new packets.

As described previously, all packets in the queues of the algorithm are classified either as overhead or non-overhead packets. Intuitively, the total value of the potential (i.e., the total number of non-overhead packets) describes the number of packets which have very limited impact on competitiveness of our algorithm (the values of the potential are at most $2n$, hence if the algorithm has only non-overhead packets it would be trivially $(0, 2n)$-competitive for the maximum load measure). Thus, in the remaining part of this section, we focus on bounding the number of overhead packets. Two possible issues may occur. First, the number of overhead packets may increase rapidly, because the adversary is allowed to inject an arbitrary number of packets in each round. However, in such case even Opt has these packets in its queues. Second, when ScanTrim recomputes thresholds at the end of some scanning cycle, if a new total threshold is lower than the current one, some of the packets may change their status from non-overhead to overhead. Showing that this occurs very rarely poses the main difficulty in our analysis.

A remark on the potential function It can be observed that when one simplifies the used potential function by choosing $S_i = \sum_{j=1}^{i}(n + 1 - j)$ instead of $S_i = \sum_{j=1}^{i} 2(n + 1 - j)$, the competitive ratio for maximum load minimization grows by a factor of $\Omega(n)$ (both in multiplicative and additive components), i.e., the resulting algorithm would be $(\Omega(n^2), \Omega(n^2))$-competitive. This demonstrates the subtlety of the chosen potential function, which is essential to control scanning and trimming processes.

More precisely, the adversary could fill the queues of the algorithm to the level of $n/2 - 1$, while keeping the queues of Opt empty (this is in fact possible for both variants of the potential). Afterwards, the adversary keeps adding packets to the last node. During this period, the algorithm operates in $\Theta(n)$ stages, each consisting of a scanning cycle and trimming of the most overloaded node (this is the node that was last in the preceding scanning cycle and has been moved to the front of $\mathcal{L}$ at the end of that cycle). In each stage, the number of packets in the last node grows by $\Theta(n)$, resulting in the maximum queue having $\Omega(n^2)$ packets, while the queues of Opt are empty (as the injection rate is 1 packet per round).

We note that our algorithm ScanTrim with the non-simplified variant of the potential (that uses $S_i = \sum_{j=1}^{i} 2(n + 1 - j)$) efficiently handles this scenario. Namely, it never runs a trimming cycle, and thus all its queues are always at most $2n$.

2.2 Semi-potentials

By the algorithm definition, at the end of a scanning cycle the list $\mathcal{L}$ becomes sorted according to the key values, and thresholds are set to the current values of the potential function. To establish a relation between the old and new values of thresholds, we want to be able to compare these potential functions. As a direct comparison might be quite complex, we introduce an auxiliary intermediate concept: a semi-potential function.

A function ψ is a semi-potential function with respect to non-negative integers $x_1, x_2, \ldots, x_n$ if the following two properties hold: (i) $0 \leq \psi(i) \leq x_i$ for any $1 \leq i \leq n$, and (ii) for any i the sum of any i values among $\psi(1), \ldots, \psi(n)$ is at most S_i. The total semi-potential is defined as $\overline{\psi} = \sum_{i=1}^{n} \psi(i)$.

Unlike in the definition of the potential function, we do not require that the values of x_i are sorted. Moreover, for a fixed sorted set of values, the potential function is defined uniquely, while there might be various semi-potential functions.

Observe that for sorted keys their potential function is also a semi-potential function. (By Fact 1, the values of potentials are non-increasing and hence the sum of an arbitrary set of i potential values is dominated by the set of i first potential values, which by the definition is at most S_i.) Moreover, it is also the "largest" possible semi-potential, as stated next.

Lemma 1. *Fix any non-negative integers $x_1, \ldots, x_n$ sorted in non-increasing order and let π be their potential. For any semi-potential ψ of these integers, it holds that $\overline{\psi} \leq \overline{\pi}$.*

Proof. We inductively show that $\sum_{j=1}^{i} \psi(j) \leq \sum_{j=1}^{i} \pi(j)$ for any $i \in \{0, \ldots, n\}$. The induction basis ($i = 0$) holds trivially. Assume that the inductive claim holds for $i < n$. Then, $\sum_{j=1}^{i+1} \psi(j) \leq \min\{S_{i+1}, x_{i+1} + \sum_{j=1}^{i} \psi(j)\} \leq \min\{S_{i+1}, x_{i+1} + \sum_{j=1}^{i} \pi(j)\} = \sum_{j=1}^{i+1} \pi(j)$. □

Lemma 1 implies the following useful property. Assume that just before sorting the keys of $\mathcal{L}$, we can construct their semi-potential of some total value. Then the total potential of these keys (after sorting) is at least that value.

2.3 Cycles of ScanTrim

The following crucial lemma is the heart of our analysis; it shows that it is possible to control the thresholds (potentials) for balanced scanning cycles.

Lemma 2. *Consider a balanced scanning cycle C in the execution of algorithm* ScanTrim. *Let π be the potential at the beginning of C. Then the total potential at the end of C is at least $\overline{\pi} + y$, where y is the number of void rounds during C.*

We defer its proof to Section 2.4 and here we show how this lemma can be used to prove the competitiveness of ScanTrim. We will compare the actual number of overhead packets to the number of packets in queues of Opt at any round. Given a particular adversarial pattern of packet injections up to some fixed time t, we denote the actual number of packets exceeding the thresholds of the algorithm ScanTrim at time t^+ by ovr_t, and the number which Opt has in queues at time t^+ by opt_t. We compute these values at time t^+ after the adversary injects packets at time t and also after the algorithm computes new values of thresholds (if it does so). Note that at time 0, all thresholds are equal to zero and as the queues of Opt and ScanTrim are equal, it holds that $\mathrm{ovr}_0 = \mathrm{opt}_0$. In the following lemma we show that the difference between the number of overhead packets and the number of packets in the queues of Opt increases only in non-balanced scanning cycles (i.e., in scanning cycles which do not result in finding sufficiently many new packets in order to switch to the trimming mode and therefore were finished with $token = n$).

Lemma 3. *Consider any scanning or trimming cycle C starting at time t and ending at time $t + r$. Then,*

1. *$ovr_{t+r} - opt_{t+r} \leq ovr_t - opt_t + n$ if C is a non-balanced scanning cycle and*
2. *$ovr_{t+r} - opt_{t+r} \leq ovr_t - opt_t$ if C is a balanced scanning cycle or a trimming cycle.*

Proof. Let π_t, π_{t+r} be the thresholds (potential functions) at the beginning of a cycle C and at the end of C, respectively. Assume that y rounds of C are void and the adversary injected s packets during C. Then, the total number of packets in queues of ScanTrim at the end of C is $\overline{\pi}_{t+r} + \mathrm{ovr}_{t+r} = \overline{\pi}_t + \mathrm{ovr}_t + s - (r - y)$, and thus $\mathrm{ovr}_{t+r} = \mathrm{ovr}_t + (\overline{\pi}_t - \overline{\pi}_{t+r}) + s + y - r$. As Opt transmits at most r packets in r rounds of C, $\mathrm{opt}_{t+r} \geq \mathrm{opt}_t + s - r$, and therefore

$$\mathrm{ovr}_{t+r} - \mathrm{opt}_{t+r} \leq \mathrm{ovr}_t - \mathrm{opt}_t + (\overline{\pi}_t - \overline{\pi}_{t+r}) + y. \tag{1}$$

If C is a balanced scanning cycle, then Lemma 2 states that $\overline{\pi}_{t+r} \geq \overline{\pi}_t + y$, and thus $\mathrm{ovr}_{t+r} - \mathrm{opt}_{t+r} \leq \mathrm{ovr}_t - \mathrm{opt}_t$.

If C is a trimming cycle, then the thresholds remained unchanged within C, by the definition of ScanTrim. That is, $\overline{\pi}_t = \overline{\pi}_{t+r}$. Moreover, ScanTrim successfully transmits one packet in each round of a trimming mode, which implies that no round is void during a trimming cycle and therefore $y = 0$. Hence, (1) implies $\mathrm{ovr}_{t+r} - \mathrm{opt}_{t+r} \leq \mathrm{ovr}_t - \mathrm{opt}_t$.

It remains to consider the case when C is a non-balanced scanning cycle. In such case $r = n$. If the i-th round of C was void, node i had zero packets at that time, which means that its threshold was zero as well. Thus, $\pi_t(i) = 0$ for at least y values of i. We define a function ψ, such that $\psi(i) = \pi_t(i) - 1$ if the i-th round of C is non-void and $\psi(i) = \pi_t(i) = 0$ otherwise. Note that for all i, it holds that $\psi(i) \leq \pi_t(i)$, and $\psi(i)$ is not larger than the key value of node i at the end of C. Therefore, ψ is a semi-potential function for key values at the end of C, and thus by Lemma 1, $\overline{\pi}_{t+r} \geq \overline{\psi}$. On the other hand, $\overline{\psi} = \overline{\pi}_t - (n - y)$ by the definition of ψ. Substituting this bound on $\overline{\pi}_{t+r}$ in (1), we obtain that $\mathrm{ovr}_{t+r} - \mathrm{opt}_{t+r} \leq \mathrm{ovr}_t - \mathrm{opt}_t + y + (n - y) \leq \mathrm{ovr}_t - \mathrm{opt}_t + n$. □

Now, we may show that ScanTrim keeps the number of overhead packets small at all queues, which will lead to the desired upper bound on its competitive ratio. Namely, we show that if the value of $\mathrm{ovr}_t - \mathrm{opt}_t$ is already large, then the subsequent cycle is either a trimming or a balanced scanning one, and therefore, by Lemma 3, this value does not grow.

Lemma 4. *For any scanning or trimming cycle C starting at time t, it holds that $ovr_t \leq opt_t + 2n$.*

Proof. We show this lemma inductively. Clearly, it is the case for the very first cycle, as $\mathrm{ovr}_0 = \mathrm{opt}_0$. Now assume that it holds for a cycle C starting at time t and ending at time $t + r$; we show it for the next cycle C' starting at time $t + r$. We use Lemma 3 to observe that:

- If C is a trimming cycle, then $\mathrm{ovr}_{t+r} - \mathrm{opt}_{t+r} \leq \mathrm{ovr}_t - \mathrm{opt}_t \leq 2n$.
- If C is a scanning cycle and $\mathrm{ovr}_t \leq \mathrm{opt}_t + n$, then $\mathrm{ovr}_{t+r} - \mathrm{opt}_{t+r} \leq \mathrm{ovr}_t - \mathrm{opt}_t + n \leq 2n$.
- If C is a scanning cycle and $\mathrm{ovr}_t > \mathrm{opt}_t + n$, then we claim that C is a balanced scanning cycle. This is clearly the case if C ended with $token < n$, so we assume that at the end of C we have $token = n$. Let x_i be the number of packets at node i at time t, and φ_i be the value of the threshold at node i during cycle C. Then, $\sum_{i=1}^{n}(x_i - \varphi_i) = \mathrm{ovr}_t > n$. Furthermore, $k_i + p_i \geq x_i$ as $k_i + p_i$ is the number of packets at node i at time $(t+i)^-$ (i.e., right before node i transmits) and between time t^+ and $(t+i)^-$ the number of packets at node i may only grow. Therefore, $\sum_{i=1}^{n}(k_i + p_i - \varphi_i) \geq \sum_{i=1}^{n}(x_i - \varphi_i) > n = token$, which shows that C is a balanced scanning cycle. In this case, $\mathrm{ovr}_{t+r} - \mathrm{opt}_{t+r} \leq \mathrm{ovr}_t - \mathrm{opt}_t \leq 2n$.

Hence, in either case the inductive claim holds. □

Theorem 1. *The algorithm* ScanTrim *is $(1, n^2 + 4n)$-competitive for the total load measure and $(n, 5n)$-competitive for the maximum load measure.*

Proof. Fix any time $t + r$ belonging to a cycle C starting at time t. By Lemma 4, $\mathrm{ovr}_t \leq \mathrm{opt}_t + 2n$. If C is a trimming cycle, then ScanTrim transmits an overhead packet in each round, i.e., the number of packet transmitted by ScanTrim is not smaller than that of Opt. Hence $\mathrm{ovr}_{t+r} \leq \mathrm{opt}_{t+r} + 2n$. If C is a scanning cycle, then its length is at most n, and thus even if ScanTrim does not transmit any overhead packets, $\mathrm{ovr}_{t+r} - \mathrm{opt}_{t+r} \leq \mathrm{ovr}_t - \mathrm{opt}_t + n$, and thus $\mathrm{ovr}_{t+r} \leq \mathrm{opt}_{t+r} + 3n$.

In either case, $\mathrm{ovr}_{t+r} \leq \mathrm{opt}_{t+r} + 3n$. The number of non-overhead packets is at most $S_n = n(n + 1)$. Therefore, the total number of packets at time $t + r$ is at most $\mathrm{opt}_{t+r} + n^2 + 4n$, which shows the first part of the theorem. Furthermore, the number of non-overhead packets at any node is at most $2n$ and hence the maximum load at time $t + r$ is at most $\mathrm{opt}_{t+r} + 3n + 2n = n \cdot (\mathrm{opt}_{t+r}/n) + 5n$. As the maximum load of Opt at time $t + r$ is at least opt_{t+r}/n, the second part of the theorem follows. □

2.4 Proof of the Crucial Lemma

In this section, we provide the proof of Lemma 2. We start with a technical claim.

Lemma 5. *Let π be a potential function and $0 < i_1 < \ldots < i_m = r$, where $0 < m \leq r \leq n$. Then, $\sum_{k=1}^{m} \pi(i_k) \leq S_m + m - r - z$, where $z = |\{j \leq r \,|\, \pi(j) = 0\}|$.*

Proof. In the proof, we extensively use the properties of Fact 1. In particular, recall that $\pi(1) \geq \cdots \geq \pi(n)$. We consider two cases.

First we show that $\sum_{k=1}^{m} \pi(i_k) \leq S_m + m - r$, i.e., a bound that is sufficient for $z = 0$. We observe that $\sum_{k=1}^{m-1} \pi(i_k) \leq \sum_{k=1}^{m-1} \pi(k)$ and $\sum_{k=m}^{r} \pi(k) \geq (r - m + 1) \cdot \pi(r)$. Hence,

$$\begin{aligned}
\textstyle\sum_{k=1}^{m} \pi(i_k) &= \textstyle\sum_{k=1}^{m-1} \pi(i_k) + \pi(r) \\
&\leq \textstyle\sum_{k=1}^{m-1} \pi(k) + \frac{1}{r-m+1} \cdot \sum_{k=m}^{r} \pi(k) \\
&= \left(1 - \frac{1}{r-m+1}\right) \cdot \textstyle\sum_{k=1}^{m-1} \pi(k) + \frac{1}{r-m+1} \cdot \sum_{k=1}^{r} \pi(k) \\
&\leq \left(1 - \frac{1}{r-m+1}\right) \cdot S_{m-1} + \frac{1}{r-m+1} \cdot S_r \\
&= S_{m-1} + 2n + 1 - r - (m-1) \\
&= S_m + m - r \ ,
\end{aligned}$$

where the last two equalities follow by Fact 2 and by the definition of S_m, respectively.

It remains to show the lemma when $z > 0$. As π are sorted in non-increasing order, $\pi(i_m) = \pi(r) = 0$. Observe that there are at most $q = \min\{m-1, r-z\}$ non-zero elements among $\pi(i_1), \ldots, \pi(i_m)$. Indeed, there are at most $m-1$ non-zero elements because $\pi(i_m) = 0$ and there are at most $r-z$ non-zero elements because there are exactly $r - z$ non-zero elements among the whole set $\pi(1), \pi(2), \ldots \pi(r)$. Hence, $\sum_{k=1}^{m} \pi(i_k) \leq \sum_{k=1}^{q} \pi(k) \leq S_q$.

It remains to show that $S_q \leq S_m + m - r - z$ or equivalently $S_m - S_q \geq r - m + z$. By Fact 2,

$$\begin{aligned}
S_m - S_q &= (m - q) \cdot (2n + 1 - m - q) \\
&\geq 2n + 1 - m - q \\
&\geq 2n + 1 - m + z - r \\
&\geq r + 1 - m + z \ ,
\end{aligned}$$

where in the last inequality we used $r \leq n$. □

Proof of Lemma 2. Let $k'_1, \ldots, k'_n$ be the values of keys at the beginning of a balanced scanning cycle C. Potential π is computed for these keys.

Let ℓ be the position (on the list $\mathcal{L}$) of the last node that transmitted during C. Let $k_1, \ldots, k_n$ be the values of keys at the end of the cycle, before they are sorted by ScanTrim and let $p_1, \ldots, p_n$ be the queue non-emptiness indicators at the end of C (also before sorting). Then,

- $k'_i \leq k_i + p_i$ for $i \leq \ell$ as $k_i + p_i$ is the number of packets at queue i right before the transmission from queue i;
- $k'_i = k_i \leq k_i + p_i$ for $i > \ell$ as there was no transmission from queue i within C.

As π is the potential for $k'_1, \ldots, k'_n$, it is also a semi-potential for values $k_1 + p_1, \ldots, k_\ell + p_\ell, k_{\ell+1}, \ldots, k_n$. We will construct a semi-potential function ψ with respect to $k_1, \ldots, k_n$, such that $\overline{\psi} = \overline{\pi} + y$. (Recall that y is the number of void rounds in C.) This will immediately conclude the proof as the total potential for the sorted values $k_1, \ldots, k_n$ is at least $\overline{\psi}$ by Lemma 1.

The function ψ is defined as

$$\psi(i) = \begin{cases} k_i & \text{for } i < \ell \ , \\ \sum_{i=1}^{\ell} \pi(i) - \sum_{i=1}^{\ell-1} k_i + y & \text{for } i = \ell \ , \\ \pi(i) & \text{for } i > \ell \ . \end{cases}$$

The relationship $\overline{\psi} = \overline{\pi} + y$ follows directly from the definition of ψ, and thus it remains to show that ψ is indeed a semi-potential function for values $k_1, \ldots, k_n$.

Since we consider a balanced scanning cycle and the threshold values are given by π, the following two properties hold (cf. Algorithm 1):

$$\sum_{i=1}^{\ell'}(k_i + p_i - \pi(i)) \leq \ell' \text{ for any } \ell' < \ell, \tag{2}$$

$$\sum_{i=1}^{\ell}(k_i + p_i - \pi(i)) > \ell \ . \tag{3}$$

We start with showing the following auxiliary relation:

$$\pi(\ell) \leq \psi(\ell) < k_\ell \ . \tag{4}$$

To this end, we observe that $y = \ell - \sum_{i=1}^{\ell} p_i$ as $\sum_{i=1}^{\ell} p_i$ is the number of non-void rounds of C. Applying it to the definition of $\psi(\ell)$ yields $\psi(\ell) = \ell + \pi(\ell) - p_\ell + \sum_{i=1}^{\ell-1}(\pi(i) - k_i - p_i)$. Thus, using (2) with $\ell' = \ell - 1$, we obtain that $\psi(\ell) \geq \ell + \pi(\ell) - p_\ell - \ell + 1 \geq \pi(\ell)$, and using (3), $\psi(\ell) = k_\ell + \ell + \sum_{i=1}^{\ell}(\pi(i) - k_i - p_i) < k_\ell + \ell - \ell = k_\ell$.

Now, it is easy to verify that ψ satisfies the first property of the semi-potential function which states that $0 \leq \psi(i) \leq k_i$ for all i. This condition holds trivially for $i < \ell$, for $i = \ell$ it follows by (4) and non-negativity of π, and for $i > \ell$ we have $\psi(i) = \pi(i) \leq k'_i = k_i$.

The second property of the semi-potential function states that the sum of any m values among $\psi(1), \ldots, \psi(n)$ is at most S_m. To show it, we fix a set $I = \{i_1, \ldots, i_m\}$, where $1 \leq i_1 < i_2 < \ldots < i_m = r \leq n$ and show that $\sum_{i \in I} \psi(i) \leq S_m$. To this end, we split the sum in question into three parts and bound separately each of them:

$$\sum_{i \in I} \psi(i) = \sum_{i \in I} \pi(i) + \sum_{i \in [1,r]} (\psi(i) - \pi(i)) + \sum_{i \in [1,r] \setminus I} (\pi(i) - \psi(i)) \ . \tag{5}$$

We also define $y_r = r - \sum_{i=1}^{r} p_i$, i.e., y_r is the number of void rounds in the first r transmissions,

Bounding the first summand. Observe that if a round i is void, then $\pi(i) \leq k'_i \leq k_i + p_i = 0$, i.e., $\pi(i) = 0$. Hence, $y_r \leq |\{j \leq r \,|\, \pi(j) = 0\}|$, and thus by Lemma 5,

$$\textstyle\sum_{i \in I} \pi(i) \leq S_m + m - r - |\{j \leq r \,|\, \pi(j) = 0\}| \leq S_m + m - r - y_r \ . \tag{6}$$

Bounding the second summand. We consider two cases. If $r \geq \ell$, the definition of ψ implies that $\sum_{i=1}^{r}(\psi(i) - \pi(i)) = y = y_r$. If $r < \ell$, (2) implies $\sum_{i=1}^{r}(\psi(i) - \pi(i)) = \sum_{i=1}^{r}(k_i - \pi(i)) \leq r - \sum_{i=1}^{r} p_i = y_r$. Therefore, in either case,

$$\textstyle\sum_{i=1}^{r}(\psi(i) - \pi(i)) \leq y_r \ . \tag{7}$$

Bounding the third summand. Recall that for $i < \ell$ it holds that $\pi(i) \leq k_i + p_i \leq k_i + 1 = \psi(i) + 1$. Furthermore, $\pi(\ell) \leq \psi(\ell)$ by (4) and $\pi(i) = \psi(i)$ for $i > \ell$ by the definition of ψ. Hence, $\pi(i) \leq \psi(i) + 1$ holds for any i. Therefore,

$$\textstyle\sum_{i \in [1,r] \setminus I} (\pi(i) - \psi(i)) \leq r - m \ . \tag{8}$$

By bounding terms in (5) using (6), (7) and (8), we immediately obtain that $\sum_{i \in I} \psi(i) \leq S_m$, which shows that ψ is a semi-potential for values $k_1, \ldots, k_n$ and concludes the proof. □

3 Lower Bounds for Online Queuing

In this section, we show that our algorithm ScanTrim is essentially optimal. First, we argue that it is optimal when we consider the total load minimization, later we show it for the maximum load case.

3.1 Total Load Minimization

Theorem 2. *For any deterministic algorithm* Alg *which is (R, A)-competitive for total load minimization, it holds that $R \geq 1$ and $A \geq (n/2-1)^2 - 1$. The bound on R holds also for randomized algorithms.*

Proof. First, we observe that $R < 1$ is not possible as the adversary may place an arbitrary number of packets in the queues at once and make the additive term A negligible.

Second, suppose for a contradiction that $A < (n/2-1)^2 - 1$. Then, by Theorem 9 of [13], the $(1,1)$-restricted adversary can cause any deterministic distributed algorithm to have the total load of at least $(n/2-1)^2$. Recall that for any given injection pattern generated by this adversary, the optimum algorithm has always the total load at most 1. Therefore, if the adversary waits one more round without injecting a packet, we get $L_{\mathrm{OPT}} = 0$ and $L_{\mathrm{ALG}} \geq (n/2-1)^2 - 1$, which contradicts the choice of A. □

3.2 Maximum Load Minimization

The lower bound for maximum load minimization is more involved. We say that an adversarial strategy $\mathcal{I}$ *clears at time* t if it is defined up to time t, exactly t packets are injected till time t (inclusively) and $L_{\mathrm{OPT}}(\mathcal{I}, t) = 0$. We do not express explicitly which adversarial strategy is considered in this notation, but it will always be clear from the context.

For a randomized algorithm Alg, let x_i be the indicator random variable that is equal to 1 if there is a collision or silence in round i and 0 otherwise. Informally speaking, round i with $x_i = 1$ under an injection strategy that clears is "wasted" by Alg. "Silence" denotes also the situation in which a node transmits a message, but this message does not contain a packet (since the queue of the transmitting node is empty).

Lemma 6. *Fix any time t, any integer B, any $\varepsilon > 0$, and an adversarial strategy $\mathcal{I}$ that clears at time t. For any randomized algorithm* Alg*, there exist time points $t'' \geq t' \geq t$ and a finite adversarial strategy $\mathcal{I}''$ that extends $\mathcal{I}$, clears at t'', and for which it holds that*

1. *either $\mathbf{E}[\sum_{i=t+1}^{t''} x_i] \geq \varepsilon/2$,*
2. *or $M_{\mathrm{OPT}}(t') = B$ and $\mathbf{E}[M_{\mathrm{ALG}}(t')] \geq (1-\varepsilon)nB$.*

Proof. Let A denote the subset $\{v_1, v_2, \ldots, v_{n-1}\}$ of all nodes except v_n. We consider an infinite extension of $\mathcal{I}$, denoted $RR(\mathcal{I})$. Namely, $RR(\mathcal{I})$ injects packets to nodes from A in round-robin fashion starting at time t, i.e., for $i \geq 0$, at round $t+i$ it injects one packet to the queue of node $v_{(i \bmod (n-1))+1}$. Note that if we cut $RR(\mathcal{I})$ after the injection of a packet at any time $\tau \geq t$, then $RR(\mathcal{I})$ clears at time $\tau + 1$.

In addition to x_i's, we define two groups of indicator random variables:

- $c_i = 1$ if no node from A transmits a packet in round i,
- $y_i = 1$ if v_n successfully transmits in round i.

For $i \geq 0$, let $X_i = \sum_{j=1}^{i} x_j$ and $X_\infty = \lim_{i\to\infty} \mathbf{E}[X_i]$. Random variables C_i, Y_i and values C_∞, Y_∞ are defined analogously.

If there is an infinite number of silent or collision rounds in the definition of $RR(\mathcal{I})$, i.e., $X_\infty = \infty$, then the first condition of the lemma holds for sufficiently large t'' and the injection sequence $RR(\mathcal{I})$ cut at time $t''-1$. Thus, in the rest of the proof we assume that $X_\infty < \infty$.

As at most t packets are injected during $\mathcal{I}$, v_n can have at most t packets at time t^+ in Alg's solution. In $RR(\mathcal{I})$ no packet is injected to v_n after time t, and therefore the number of successful transmissions from node v_n is finite and bounded by t. That is, $Y_i \leq t$ for any i, and hence $Y_\infty \leq t < \infty$.

Fix any round i. If no node from A transmits at round i, it implies either silence or a successful transmission by v_n. Hence, $c_i \leq x_i + y_i$ and therefore $C_\infty \leq X_\infty + Y_\infty < \infty$. This means that the number of rounds in which no node from A transmits is finite. The intuition behind the rest of the proof is as follows. First, we pick an appropriate time r, such that after this time ALG transmits essentially only from A. Then we modify $RR(\mathcal{I})$ by injecting nB packets to v_n. From a perspective of nodes from A, the newly created strategy is indistinguishable from $RR(\mathcal{I})$. Therefore, to avoid discrepancy between its maximum loads of ALG and OPT, v_n has to signal that it has new packets, in the only way possible, that is by transmitting them. The latter action inevitably creates a collision.

Formally, let $r \geq t$ be such that $\mathbf{E}[C_r] \geq C_\infty - \varepsilon/2$. Let $t' = r + (n-1)B$ and $t'' = r + (2n-1)B$. Let $\mathcal{E}$ be the event that $\sum_{i=r+1}^{t'} c_i = 0$, i.e., that at least one element of $A = \{v_1, \ldots, v_{n-1}\}$ transmits in each round from $r+1, \ldots, t'$ under strategy $RR(\mathcal{I})$. Then

$$C_\infty \geq \mathbf{E}[C_{t'}] \geq \mathbf{E}[C_r] + \Pr[\neg\mathcal{E}] \geq C_\infty - \varepsilon/2 + \Pr[\neg\mathcal{E}] \ ,$$

and therefore $P[\neg\mathcal{E}] \leq \varepsilon/2$, i.e., $P[\mathcal{E}] \geq 1 - \varepsilon/2$,

Now, we analyze the behavior of ALG under a following modification of the strategy $RR(\mathcal{I})$, denoted $\mathcal{I}''$, which is a finite strategy ending at time t''. Namely, at time r, the adversary additionally injects $n \cdot B$ packets to the queue of v_n and at time points $t', \ldots, t''$ no packets are injected. Note that $\mathcal{I}''$ clears at t''. We now show that $\mathcal{I}''$ satisfies the conditions of the lemma.

We look at the actions of ALG for node v_n in rounds $r+1, \ldots, t'$ of $\mathcal{I}''$. Let F be the event that v_n transmits at least once within these rounds (under strategy $\mathcal{I}''$). We consider two cases, depending on the value of $\Pr[F]$.

1. $\Pr[F] \leq \varepsilon$. We analyze the state of queues at time $(t')^+$ under event $\neg F$ (v_n does not transmit in rounds $r+1, \ldots, t$). Independently of actions of nodes from A, it holds that $M_{\mathrm{ALG}}(t') \geq n \cdot B$ as the load at v_n is at least $n \cdot B$. As $\neg F$ occurs with probability at least $1 - \varepsilon$, $\mathbf{E}[M_{\mathrm{ALG}}(t')] \geq (1-\varepsilon)nB$. On the other hand, OPT can transmit $(n-1)B$ packets from v_n in rounds $r+1, \ldots, t'$, ending time t' with all queues of size B, i.e. $M_{\mathrm{OPT}}(t') = B$. Hence, the second condition of the lemma follows in this case.

2. $\Pr[F] > \varepsilon$. We prove that with probability at least $\varepsilon/2$, there is at least one collision within rounds $r+1, \ldots, t'$, and hence also within rounds $t+1, \ldots, t''$. For this part of the proof, let τ be the random variable denoting the first round from $r+1, \ldots, t'$ in which v_n transmits, and equal to t' if v_n does not transmit within these rounds. In these terms, the event F means that v_n actually transmits at round τ. We also define event $\mathcal{E}'$ denoting that at least one node from A transmits in each round from $r+1, \ldots, \tau$. In rounds $r+1, \ldots, \tau$ (inclusively) no node from A can distinguish between strategies $RR(\mathcal{I})$ and $\mathcal{I}''$ and hence $\mathcal{E}' \supseteq \mathcal{E}$, which means that $\Pr[\mathcal{E}'] \geq \Pr[\mathcal{E}] \geq 1 - \varepsilon/2$. The event $\mathcal{E}' \cap F$ implies that there is a collision at round τ, and such event occurs with probability $\Pr[\mathcal{E}' \cap F] \geq \Pr[\mathcal{E} \setminus \neg F] \geq \Pr[\mathcal{E}'] - \Pr[\neg F] \geq (1 - \varepsilon/2) - (1 - \varepsilon) = \varepsilon/2$. Hence, the first condition of the lemma follows in this case.

□

Theorem 3. *For any deterministic algorithm* ALG *which is (R, A)-competitive for maximum queue minimization, it holds that $R \geq n$ and $A = \Omega(n)$. The bound on R holds also for randomized algorithms.*

Proof. We start from zero-length adversarial strategy that does not inject anything and iteratively apply Lemma 6. This way, either we generate a strategy that clears at some time t ($M_{\mathrm{OPT}}(t) = L_{\mathrm{OPT}}(t) = 0$) for which $\mathbf{E}[\sum_{i=1}^{t} x_i]$ is arbitrarily high (and hence the maximum load of ALG is arbitrarily high), or we find a step t' such that $\mathbf{E}[M_{\mathrm{ALG}}(t')] \geq (1-\varepsilon)Bn = (1-\varepsilon)n \cdot M_{\mathrm{OPT}}(t')$. Since B may be chosen arbitrarily large and ε arbitrarily small, it diminishes the importance of the additive constant A and proves that $R \geq n$.

In order to provide a lower bound for A, recall that by Theorem 2 it is possible to create an adversarial strategy that leads to $L_{\mathrm{OPT}} = 0$ and $L_{\mathrm{ALG}} \geq (n/2)^2 - 1$. At such time, it also holds that $M_{\mathrm{OPT}} = 0$ and $M_{\mathrm{ALG}} \geq L_{\mathrm{ALG}}/n = \Omega(n)$. □

4 Stability of Scan-Trim in the Stochastic Model

As defined in the introduction, we now assume a stochastic model in which packets are injected according to a random distribution, defined by the sequence of numbers $p_1, \ldots, p_n \in (0, 1)$. In each round, for each node independently, one packet is injected into the queue of node i with probability p_i and no packet is injected to this queue with probability $1 - p_i$. Our goal is to analyze the total load of *deterministic* distributed algorithms in such a scenario.

4.1 Stability of the Centralized Solution

First, we focus on the centralized algorithm Opt which has the full knowledge on the queue sizes and chooses an arbitrary (and exactly one) packet to be transmitted in each round, provided at least one queue is not empty. As Opt is centralized, the actual distribution of packets is unimportant, and we simply analyze its total load. We want to investigate the conditions sufficient and necessary for reducing the total load to zero, i.e., emptying all queues.

The evolution of the Opt's total load can be described by a time-homogeneous Markov chain (also denoted Opt) whose states are non-negative integers corresponding to the total load. Furthermore, for any time point $i \geq 0$, S_i is the random variable being the state of Opt at time i; in particular, S_0 is the initial number of packets in all queues. For any time-homogeneous Markov chain C and any two states S and S', we denote the probability that C ever reaches state S' when it starts from S by $P_C(S \to S')$ and the expected number of steps to hit S' for the first time by $\mathbf{E}_C(S \to S')$.

We are interested in the event of reaching the empty queues state, and therefore we focus on estimating the values of $P_C(S_0 \to 0)$ and $\mathbf{E}_C(S_0 \to 0)$. In these terms, stability means that $P_C(S_0 \to 0) = 1$ for any initial state S_0 and strong stability means that $\mathbf{E}_C(S_0 \to 0) < \infty$. Note that strong stability trivially implies stability.

The goal of this section is to present tight conditions on $\sum_{i=1}^{n} p_i$ that ensure stability and strong stability of Opt. To this end, let Z_t be the random variable denoting the number of packets injected at time t. Recall that by the definition of our process, all Z_t are identically and independently distributed, their support is the set $\{0, \ldots, n\}$ and their mean is equal to $\sum_{i=1}^{n} p_i$. As Opt transmits a packet in round t if it has any ($S_{t-1} > 0$), the transitions between consecutive states are defined by

$$S_t = \begin{cases} S_{t-1} + Z_t - 1 & \text{if } S_{t-1} > 0 \ , \\ S_{t-1} + Z_t & \text{if } S_{t-1} = 0 \ . \end{cases} \tag{9}$$

Clearly, the Markov chain Opt is irreducible as for any two states S and S' and large enough τ, there is a positive probability that Opt changes state from S to S' within τ steps. As there is a positive probability that the number of packets remains the same (i.e., the state does not change), Opt is also aperiodic.

Opt is a particular case of a *Repair Shop* Markov chain (see page 59 of [11]): this chain is also defined by (9), but allows any i.i.d. random variables Z_t of support $\{0, 1, \ldots, n\}$, while Z_t are sums of n independent Bernoulli variables in the case of Opt. By the results for the Repair Shop Markov chain (see page 188 of [11]), it follows that:

- if $\sum_{i=1}^{n} p_i < 1$, Opt is positive recurrent, i.e., $\mathbf{E}_{\mathrm{OPT}}(S \to S) < \infty$ for any state S;
- if $\sum_{i=1}^{n} p_i = 1$, Opt is null recurrent, i.e., $P_{\mathrm{OPT}}(S \to S) = 1$ and $\mathbf{E}_{\mathrm{OPT}}(S \to S) = \infty$ for any state S;
- if $\sum_{i=1}^{n} p_i > 1$, Opt is transient, i.e., $P_{\mathrm{OPT}}(S \to S) < 1$ for any state S.

As an irreducible Markov chain can be either positive recurrent, null recurrent or transient, the above three conditions are both sufficient and necessary. An irreducible chain is recurrent if it is positive recurrent or null recurrent. Thus, OPT is recurrent if and only if $\sum_{i=1}^{n} p_i \leq 1$.

Lemma 7. *If $\sum_{i=1}^{n} p_i \leq 1$, then Opt is stable. Moreover, if $\sum_{i=1}^{n} p_i < 1$, then Opt is strongly stable.*

Proof. We fix any starting state S_0.

If $\sum_{i=1}^{n} p_i \leq 1$, then OPT is recurrent, and thus $P_{\text{OPT}}(S_0 \to S_0) = 1$. Then $P_{\text{OPT}}(S_0 \to 0) = 1$ follows immediately by irreducibility of OPT.

If $\sum_{i=1}^{n} p_i < 1$, then OPT is positive recurrent, and thus $\mathbf{E}_{\text{OPT}}(S_0 \to S_0) < \infty$. For a positive recurrent Markov chain on a countable set of states it holds that $\mathbf{E}_{\text{OPT}}(S_0 \to 0) + \mathbf{E}_{\text{OPT}}(0 \to S_0) = E_{\text{OPT}}(S_0 \to S_0)/\theta(S_0, 0)$, where $\theta(S, S')$ denotes the probability that OPT, starting from S, reaches state S' at least once before returning to S [19]. As OPT is irreducible, $\theta(S_0, 0) > 0$, and therefore $\mathbf{E}_{\text{OPT}}(S_0 \to 0) < \infty$, which concludes the proof. □

We note that the implications in Lemma 7 are in fact equivalences. Assume that $P_{\text{OPT}}(S_0 \to 0) = 1$ for all states $S_0 \geq 0$, i.e., OPT is stable. As this condition holds also for $S_0 = 0$, state 0 is recurrent. By irreducibility, the chain is recurrent, which implies $\sum_{i=1}^{n} p_i \leq 1$. In the same way, we may show that if $\mathbf{E}_{\text{OPT}}(S_0 \to 0) < \infty$ for all states $S_0 \geq 0$, then the chain is positive recurrent, which implies $\sum_{i=1}^{n} p_i < 1$.

4.2 Stability of the Distributed Solution

Now, we move our attention to studying stability and strong stability of online distributed (deterministic) algorithms.

Lemma 8. *Fix any monotonic functions $f, g : \mathbb{N} \to \mathbb{N}$. Assume that a deterministic distributed online algorithm* ALG *is $(1, f(n))$-competitive with respect to total load. Assume that given arbitrary placement of m packets in its queues,* ALG *transmits them all in the next $g(m)$ rounds, provided no packet is injected in this period. If* OPT *is stable, then so is* ALG*.*

Proof. For the proof, we run OPT and ALG on the same input sequence. Assume that OPT has empty queues at time t and there is no packet injection at time points $t+1, t+2, \ldots, t+g(f(n))$. The competitiveness of ALG implies that it has at most $f(n)$ packets in its queues at time t^+, and thus at time $(t+g(f(n)))^+$ it has no packet. As OPT is stable, it empties its queues infinitely many times with probability 1. We show that after some of those time points there exist a period of length $g(f(n))$ during which nothing is injected.

To formalize this intuition, we consider the stochastic process of injecting packets. It defines the probability space Ω, whose elements are infinite injection patterns (called patterns for short). With each element of Ω, we associate a sequence of states of OPT, describing the number of packets OPT has on such injection pattern at particular time points.

For any time t, we define an event I_t denoting that at least one packet is injected at time points $t+1, t+2, \ldots, t+g(f(n))$. Note that $\Pr[I_t] = 1 - q^{g(f(n))}$, where $q > 0$ is the probability that nothing is transmitted in a single round. In other words, $\Pr[I_t]$ is a constant smaller than 1 and independent of t, which we will denote by Q. We say that *zero occurs* at time $t \geq 0$ if OPT is in state 0 at time t. We call a zero occurring at time t *weak* if additionally event I_t occurs, and *strong* otherwise.

We partition Ω into three disjoint sets: F: the set of all injection patterns that have finitely many zeros, T: the set of all injection patterns that have infinitely many zeros and at least one of these zeros is strong, and W: the set of all injection patterns that have infinitely many zeros and all these zeros are weak.

As already mentioned at the beginning of the proof, for any injection pattern from set T, there is a time at which ALG has empty queues. Therefore, it is sufficient to show that the probability measure of set T is 1, or alternatively that probability measures of sets F and W are equal to zero. $\Pr[F] = 0$ follows immediately by the stability of OPT.

For showing $\Pr[W] = 0$, for any $k \geq 0$ and time $j \geq 0$, we define the event R_k^j denoting that (i) there are exactly $k \cdot g(f(n))$ zeros at time points $0, 1, 2, \ldots, j-1$, (ii) for $0 \leq i \leq k-1$ the $(i \cdot g(f(n)) + 1)$-st zero is weak (the remaining zeros may be strong or weak), and (iii) there is (the $k \cdot (g(f(n)) + 1)$-st) zero (strong or weak) at time j. Note that

$$W \subseteq \bigcap_{k=1}^{\infty} \bigcup_{j=0}^{\infty} R_k^j \ ,$$

because the right hand side describes patterns that have infinitely many zeros, such that the $(i \cdot g(f(n)) + 1)$-st zero is weak for $i \geq 0$. Another important property (that clarifies the cumbersome definition of R_k^j) is that

R_k^j depends only on the injections up to time j as the last zero for which we require that it is weak occurs no later than at the position $j - g(f(n))$.

In the following, we build a recursive bound on $\Pr[\bigcup_{j=0}^{\infty} R_k^j]$ for any k. As $\bigcup_{j=0}^{\infty} R_0^j$ is the event of having at least one zero in the sequence, $\Pr[\bigcup_{j=0}^{\infty} R_0^j] = 1$ by the stability of Opt. Next, we observe that

$$\bigcup_{j=1}^{\infty} R_{k+1}^j \subseteq \bigcup_{j=1}^{\infty} \left(R_k^j \cap I_j \right) \ .$$

This follows as the left side denotes the set containing all patterns having at least $k \cdot g(f(n)) + 1$ zeros, while the right side the set with patterns having at least $(k-1) \cdot g(f(n)) + 1$ zeros, and for both left and right side, the $(i \cdot g(f(n)) + 1)$-st zeros are weak for $0 \leq i \leq k-1$. Therefore,

$$\begin{aligned}
\Pr\left[\bigcup_{j=1}^{\infty} R_{k+1}^j\right] &\leq \Pr\left[\bigcup_{j=1}^{\infty} \left(R_k^j \cap I_j \right)\right] \\
&\leq \sum_{j=1}^{\infty} \Pr\left[R_k^j \cap I_j\right] = \sum_{j=1}^{\infty} \Pr[R_k^j] \cdot \Pr\left[I_j\right] \\
&= Q \cdot \sum_{j=1}^{\infty} \Pr\left[R_k^j\right] = Q \cdot \Pr\left[\bigcup_{j=1}^{\infty} R_k^j\right] .
\end{aligned}$$

The first equality follows by the independence of R_k^j and I_j and the last one because R_k^j and R_k^i are disjoint for $i \neq j$. By a simple induction, $\Pr[\bigcup_{j=1}^{\infty} R_k^j] \leq Q^k$ for any $k \geq 0$, and thus

$$\Pr[W] = \Pr\left[\bigcap_{k=1}^{\infty} \bigcup_{j=0}^{\infty} R_k^j\right] = \lim_{k \to \infty} \Pr\left[\bigcup_{j=1}^{\infty} R_k^j\right] \leq \lim_{k \to \infty} Q^k = 0 \ .$$

This finishes the proof. □

Lemma 9. *Fix any monotonic functions $f, g : \mathbb{N} \to \mathbb{N}$. Assume that a deterministic distributed online algorithm* Alg *is $(1, f(n))$-competitive with respect to total load. Assume that given arbitrary placement of m packets in its queues,* Alg *transmits them all in the next $g(m)$ rounds, provided no packet is injected in this period. If* Opt *is strongly stable, then so is* Alg*.*

Proof. We define the event I_t, value of Q, and the notions of strong and weak zeros exactly as in the proof of Lemma 8. In these terms, we need to show that for any initial state S_0, the expected time until the occurrence of a strong zero is finite.

Assume that at some time t a zero occurs. Let K be the random variable denoting the number of additional rounds till the occurrence of a strong zero. With probability $1 - \Pr[I_t]$, the zero at time t is already strong and then $K = 0$. With the remaining probability, the zero at time t is weak. As at most n packets are injected in a single time, the total number of packets in Opt queues at time $(t + g(f(n)))^+$ is at most $n \cdot g(f(n))$. Thus, the additional expected number of rounds till Opt reaches state zero again is $m := \max_{j \leq n \cdot g(f(n))} \{\mathbf{E}_{\mathrm{OPT}}(j \to 0)\}$. Afterwards, in expectation, we still need $\mathbf{E}[K]$ rounds. Hence,

$$\mathbf{E}[K] \leq (1 - \Pr[I_t]) \cdot 0 + \Pr[I_t] \cdot (g(f(n)) + m + \mathbf{E}[K]) \ , \tag{10}$$

Solving (10) for $\mathbf{E}[K]$ and substituting $\Pr[I_t] = Q$ yields $\mathbf{E}[K] \leq (g(f(n)) + m) \cdot Q/(1-Q)$.

Finally, we observe that $\mathbf{E}_{\mathrm{ALG}}(S_0 \to 0) = \mathbf{E}_{\mathrm{OPT}}(S_0 \to 0) + \mathbf{E}[K]$. As Opt is strongly stable, $\mathbf{E}_{\mathrm{OPT}}(S_0 \to 0) < \infty$ and $m < \infty$. As Q is a well defined constant, $\mathbf{E}[K]$ is finite and the lemma follows. □

Theorem 4. ScanTrim *is stable for $\sum_{i=1}^{n} p_i \leq 1$ and strongly stable for $\sum_{i=1}^{n} p_i < 1$.*

Proof. By Lemma 7, Opt is stable for $\sum_{i=1}^{n} p_i \leq 1$ and strongly stable for $\sum_{i=1}^{n} p_i < 1$. Hence, it is sufficient to fix the functions f and g satisfying the conditions of Lemma 8 and Lemma 9.

By Theorem 1, $f(n) = O(n^2)$. Now assume that ScanTrim has m packets in its queues and no subsequent packet is injected. Note that in a trimming cycle, a packet is sent in each round. Furthermore, at least one packet is transmitted in a scanning cycle (which consists of at most n rounds). Therefore, all m packets are transmitted in at most $g(m) = O(n \cdot m)$ rounds. □

5 Conclusion and Open Problems

We studied competitiveness of deterministic distributed algorithms with respect to two important performance measures: total and maximum load. Our solution is asymptotically optimal with respect to both measures. All our competitive results regarding distributed environment can be contrasted with centralized online queuing, and the obtained picture suggests that there is no simple way of transforming centralized online algorithms into distributed ones. We also show a transformation from the world of competitive analysis of distributed online queuing into distributed stochastic queuing.

This work opens several new directions in the area of online distributed queuing on a multiple-access channel. For instance:

- analysis of other performance measures, e.g., related to timing or energy efficiency,
- studying similar online distributed frameworks for streaming of dependable packets, messages of different sizes or deadlines,
- considering channel with dependencies, e.g., SINR.

Acknowledgments

We thank Dariusz Buraczewski for helpful discussions on the stochastic stability and anonymous reviewers for a number of suggestions that helped to improve the presentation.

References

[1] Miklós Ajtai, James Aspnes, Cynthia Dwork, and Orli Waarts. A theory of competitive analysis for distributed algorithms. In *35th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 401–411, Santa Fe, New Mexico, USA, 1994. IEEE.

[2] Lakshmi Anantharamu, Bogdan S. Chlebus, Dariusz R. Kowalski, and Mariusz A. Rokicki. Deterministic broadcast on multiple access channels. In *Proc. 29th IEEE International Conference on Computer Communications (INFOCOM)*, pages 146–150, San Diego, CA, USA, 2010. IEEE.

[3] Lakshmi Anantharamu, Bogdan S. Chlebus, and Mariusz A. Rokicki. Adversarial multiple access channels with individual injection rates. *Theory Comput. Syst.*, 61(3):820–850, 2017.

[4] Matthew Andrews, Baruch Awerbuch, Antonio Fernández, Frank Thomson Leighton, Zhiyong Liu, and Jon M. Kleinberg. Universal-stability results and performance bounds for greedy contention-resolution protocols. *J. ACM*, 48(1):39–69, 2001.

[5] Antonio Fernández Anta, Miguel A. Mosteiro, and Jorge Ramón Muñoz. Unbounded contention resolution in multiple-access channels. *Algorithmica*, 67(3):295–314, 2013.

[6] James Aspnes. Competitive analysis of distributed algorithms. In Amos Fiat and Gerhard J. Woeginger, editors, *Online Algorithms, The State of the Art*, pages 118–146. Springer, New York, 1998.

[7] Amotz Bar-Noy, Ari Freund, Shimon Landa, and Joseph Naor. Competitive on-line switching policies. *Algorithmica*, 36(3):225–247, 2003.

[8] Giuseppe Bianchi. Performance analysis of the IEEE 802.11 distributed coordination function. *IEEE Journal on Selected Areas in Communications*, 18:535–547, 2000.

[9] Allan Borodin and Ran El-Yaniv. *Online computation and competitive analysis.* Cambridge University Press, New York, NY, USA, 1998.

[10] Allan Borodin, Jon M. Kleinberg, Prabhakar Raghavan, Madhu Sudan, and David P. Williamson. Adversarial queuing theory. *J. ACM*, 48(1):13–38, 2001.

[11] Pierre Brémaud. *Markov chains: Gibbs fields, Monte Carlo simulation, and queues.* Springer-Verlag, Berlin; New York, 1999.

[12] Bogdan S. Chlebus. Randomized communication in radio networks. *Handbook on Randomized Computing, P.M. Pardalos, S. Rajasekaran, J.H. Reif, and J.D.P. Rolim, (Eds.), Kluwer Academic*, I:401–456, 2001.

[13] Bogdan S. Chlebus, Dariusz R. Kowalski, and Mariusz A. Rokicki. Maximum throughput of multiple access channels in adversarial environments. *Distributed Computing*, 22(2):93–116, 2009.

[14] Bogdan S. Chlebus, Dariusz R. Kowalski, and Mariusz A. Rokicki. Adversarial queuing on the multiple access channel. *ACM Trans. Algorithms*, 8(1):5, 2012.

[15] Gianluca De Marco and Dariusz R. Kowalski. Fast nonadaptive deterministic algorithm for conflict resolution in a dynamic multiple-access channel. *SIAM J. Comput.*, 44(3):868–888, 2015.

[16] Rudolf Fleischer and Hisashi Koga. Balanced scheduling toward loss-free packet queuing and delay fairness. *Algorithmica*, 38:363–376, 2004.

[17] Robert G. Gallager. A perspective on multiaccess channels. *IEEE Transactions on Information Theory*, 31(2):124–142, 1985.

[18] Leslie Ann Goldberg. Notes on contention resolution. Available online, 2000.

[19] T. E. Harris. First passage and recurrence distributions. *Transactions of the American Mathematical Society*, 73:471–486, 1952.

[20] Johan Håstad, Frank Thomson Leighton, and Brian Rogoff. Analysis of backoff protocols for multiple access channels. *SIAM J. Comput.*, 25(4):740–774, 1996.

[21] Andréa W. Richa, Christian Scheideler, Stefan Schmid, and Jin Zhang. Competitive throughput in multi-hop wireless networks despite adaptive jamming. *Distributed Computing*, 26(3):159–171, 2013.