Distributed Online and Stochastic Queuing on a Multiple Access Channel[⋆]

Marcin Bienkowski[1], Tomasz Jurdzinski[1,4], Miroslaw Korzeniowski[2,3], and Dariusz R. Kowalski[4]

[1] Institute of Computer Science, University of Wrocław, Poland
[2] Inst. of Mathematics and Computer Science, Wrocław Univ. of Technology, Poland
[3] LaBRI, Univeristy of Bordeaux 1, France
[4] Department of Computer Science, University of Liverpool, UK

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 each round, an arbitrary number of packets can be injected into the system, each to an arbitrary node's queue. Two measures of performance are considered: the total number of packets in the system, called the total load, and the maximum queue size, called the maximum load. In the online setting, we develop a deterministic 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 then the total load of any other algorithm, including centralized offline solutions, by only $O(n^2)$, 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. Next, we show that our algorithm is stochastically optimal for *any* expected injection rate smaller or equal to 1. To the best of our knowledge, this is the first solution to the stochastic queuing problem on a multiple access channel that achieves such optimality for the (highest possible) rate equal to 1.

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.

[⋆] Supported by MNiSW grant number N N206 368839, 2010-2013. The work of the second and the fourth author was supported by the Engineering and Physical Sciences Research Council [grant number EP/G023018/1]. The third author is supported by grant 2010/342643 of the Institute of Mathematics and Computers Science of the Wroclaw University of Technology.

Distributed queuing on a multiple access channel is one of the most fundamental problems, widely studied by both theoreticians (cf. [7]) and practitioners (cf. [4]). 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 injection patterns and for stochastic injections. The best up-to-date results guarantee bounded queues only for *arbitrarily bounded* packet burst, in case of the former setting, and only for stochastic injection rates *smaller* than 1 in the latter one. This work aims to resolve the remaining cases of heavy traffic in affirmative. We believe that the newly developed and analyzed distributed scheduling techniques could substantially improve flow stability in heavy traffic systems, such as data and video streaming under 802.11aa.

The Model. We consider the scenario where n nodes with pairwise disjoint 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 times are numbered from 0. At time 0, the adversary may inject 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 times $t-1$ and 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 arbitrary number of packets, each placed into an arbitrary node queue.

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 work 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 we argue later that in a single message 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, as proved in [8] even for restricted adversaries and $n \geq 3$.

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 bounded burst injection patterns have been considered so far, without comparison to the optimal solution. We describe the competitive approach in the remainder of this section. The detailed description of the stochastic queuing setting is deferred to Section 3.

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

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 step t it holds that $L_{\text{ALG}}(\mathcal{I}, t) \leq R \cdot L_{\text{OPT}}(\mathcal{I}, t) + A$ where OPT is the optimal *offline centralized* solution for injection pattern $\mathcal{I}$ until round t.

We call a randomized online distributed algorithm (R, A)-competitive for minimizing the total load if for any adversarial pattern of packet injections $\mathcal{I}$ and any time step t it holds that $\mathrm{E}[L_{\text{ALG}}(\mathcal{I}, t)] \leq R \cdot L_{\text{OPT}}(\mathcal{I}, t) + A$, where the expectation is taken over all random choices of the algorithm up to the step t.

For both deterministic and randomized algorithms we define competitiveness for minimizing maximum load in analogous way. We emphasize that the relation between ALG and OPT has to hold for *any* step t and *any* injection pattern $\mathcal{I}$. Note that, unlike in the traditional approach of competitive analysis [5], we explicitly give the additive factor in the competitive ratio.

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, in a competitive way, two important complexity measures: total load and max-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 results for stochastic optimality of protocols, so far obtained only for injection rates smaller than 1.

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., transmitting 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 [10], and independently Bar-Noy et al. [3], proved that any algorithm serving always a longest nonempty queue achieves asymptotically optimal competitive ratio of $\Theta(\log n)$, including also randomized solutions. Fleischer and Koga [10] proved additionally that the popular round-robin algorithm is $\Omega(m)$-competitive, where m is the number of injected packets. Note that the latter result qualifies as non-competitive in case of unbounded number of injected packets. The comparison of results for the centralized model and the distributed one, obtained in this work, is given in Table 1. In particular, the discrepancy between the results in these two models shows that the lack of central coordination tremendously 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

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

	centralized		distributed	
minimizing total load	OPT	(straightforward)	OPT + $\Theta(n^2)$	(this paper)
minimizing maximum load	$\Theta(\log n) \cdot$ OPT	[3,10]	$n \cdot$ OPT + $O(n)$	(this paper)

networks [2,6], several papers analyzed *distributed* queuing on a multiple access channel under *restricted* adversarial injection patterns. Previous works by Chlebus et al. [8] and Anantharamu et al. [1] considered adversaries that were (ρ, b)-restricted, also called (ρ, b)-leaky-bucket, for $\rho \leq 1$ and fixed $b \geq 1$. The restriction is that in each time interval I, the adversary may only inject $\rho \cdot |I| + b$ packets into the system. Moreover, the solutions were analyzed in a worst-case manner with respect to parameters n, ρ, b. Restricted adversaries were also used for modelling jamming on a multiple access channel [14].

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 some periods of "burstiness" growing arbitrarily large that were not allowed by the restricted adversary. Moreover, the previous results for the restricted adversary provided only global bounds on queue sizes in case they were bounded, while competitive analysis provided in this work compares the solution to the optimal algorithm at any single round. Although algorithms designed and analyzed under the restricted adversarial injection patterns may not imply similar results in more general competitive model considered in this work, some lower bounds can be adopted. In particular, Chlebus et al. [8] 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.

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 [12] and Chlebus [7] for an overview of early and middle-stage research. In particular, Håstad et al. [13] proved stochastic optimality of polynomial backoff protocols for any fixed injection rate smaller than 1 and disproved it in case of exponential backoff. To the best of our knowledge, all the previous results concerning stochastic optimality were proved for expected fixed injection rates *strictly smaller* than 1. Ours is the first deterministic distributed online algorithm achieving stochastic optimality also for (highest possible) injection rate 1.

1.2 Our Results

We develop a deterministic distributed online algorithm SCAT, whose competitiveness is asymptotically optimal with respect to both the total number of packets in the system and the maximum queue size.

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

That is, the total load of our algorithm is, in each round, larger by an additive factor 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 factor $O(n)$. The optimality of both bounds is justified by the following results, the former holding even for randomized algorithms.

Theorem 2. *For any randomized algorithm* Alg *which is* (R, A)*-competitive for maximum queue minimization, it holds that* $R \geq n$.

Theorem 3. *For any deterministic algorithm* Alg *which is* (R, A)*-competitive for total load minimization, it holds that* $R \geq 1$ *and* $A \geq (n/2 - 1)^2 - 1$.

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 time of an execution of a protocol (which might be arbitrarily large).

Furthermore, we show efficiency of our algorithm with respect to the *stochastic queuing problem* in a distributed setting for expected injection rate 1. More precisely, we show that our algorithm reaches the state with empty queues infinitely many times with probability 1, regardless of the initial distribution of packets, provided packets are injected to the system randomly according to the Bernoulli distribution with the expected number of 1 packet per round. (Note that rate 1 is the highest possible to obtain so defined stochastic optimality.) All previous solutions to this variant of the problem guaranteed such property only for the expected number of packets per round *strictly smaller* than 1. For this case, we show even a stronger property: that the expected number of steps needed to reach the state with empty queues is finite.

Due to space limit, omitted proofs will appear in the full version of the paper.

Distributed Online Solution: Challenges and Ideas. Our main online algorithm Scat is designed to overcome two fundamental challenges imposed by the shared channel: delay in updating information (there is at most one node transmitting successfully at a time, therefore a common knowledge about majority of nodes come from $\Omega(n)$ rounds in the past), and waste (i.e., collision or silence) caused during information gathering or otherwise by scheduling packets without fairly accurate information.

To demonstrate these problems, 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 some pre-defined list. It generates an unbounded waste when the adversary injects all packets to a single node, one packet per round. One could modify this protocol to empty the whole queue where visiting a node, which would prevent such waste as considered before. However, an unbounded waste is obtained in a slightly more sophisticated scenario when 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 buffer to amortize the waste generated by checking the queues of potentially empty node: such an idea was introduced by Chlebus et al. [8], who proposed algorithm Move-Big-To-Front (MBTF for short). In this algorithm, the round robin procedure is applied until a node with queue larger than n is found; in such case, the queue is moved into 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, and then—by injecting packets always to the last node on the list—create queues of size $\Omega(n^2)$, while the optimum solution has at most one packet in the whole system at each round.

The above examples are token-based protocols. Bianchi [4] 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 ad hoc transmission pattern may cause even more waste comparing 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 waste caused by silent rounds. More precisely, the algorithm runs in two modes: scanning and trimming. The former is to update the information, the latter is to transmit packets so to compete with the optimal solution. The potential function defines the order of scanned and trimmed nodes, and conditions when to switch between the two modes (i.e., efficiently between the progress in information update and in keeping the queues balanced).

The result in the stochastic injection setting is obtained by proving (positive) recurrence of the underlying Markov chains in two steps. First, we define and analyze some idealistic Markov chains, corresponding to the behavior of offline solutions. In particular, we prove that properties of these Markov chains imply 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 satisfies stochastic optimality.

2 Competitive Algorithm SCAT

In this section, we show a protocol Scan-And-Trim (Scat) and prove that it is $(n, O(n))$-competitive. We start with a high-level description, accompanied by intuitions. Then, we provide the pseudo-code and a sketch of the analysis.

The number of nodes n and the id of a node are the only input parameters for the algorithm executed by 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.

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*. Note that this is for notational simplicity, as we may assume that not transmitting anything has the same semantics.

Our algorithm abstracts from the 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, and the list of all the nodes augmented with additional data. 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. Recall that the protocol is collision free, and therefore the information heard by all nodes is the same.

Hence, we call these variables *global*, keeping in mind that, in fact, they are stored locally, but coherence between these variables' values is ensured. We also emphasize that except for its own packet queue, no node holds any other non-global 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 j", we mean the node with the j-th position on list $\mathcal{L}$. Additionally, $\mathcal{L}$ stores three pieces of information for each node q:

- Key, equal to the queue size of q attached to the last message transmitted by q on the channel; the queue size is computed without the tranmitted packet. If q has not yet transmitted, the key is set to zero;
- Threshold value for q, equal to a non-negative integer, whose value will be determined later (and modified only at some particular rounds of the protocol).
- Queue non-emptiness indicator equal to 1 when the queue of q 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.

The key, the threshold, and the indicator of the i-th element on the list are denoted k_i, $\varphi(i)$, and p_i, respectively. 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*;

The main problem the algorithm has to cope with is the information delay: the keys stored in $\mathcal{L}$ are usually outdated: the nodes do not have the information about the recent changes to the queues made by packet injections. Instead, $\mathcal{L}$ contains information about the queue sizes of a specific node from the time this node last transmitted. A great advantage of the global variables—representing only a partial knowledge of the system—is that they allow for more coordinated approach, which is easier to analyze. It appears that such approach is sufficient to achieve good performance.

Potentials. 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

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

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 $\overline{\pi} = \sum_{i=1}^{n} \pi(i)$.

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

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

Lemma 1. *Let π be a potential function and $0 < i_1 < \ldots < i_\ell = p$, where $0 < p, \ell \le n$. Then, $\sum_{k=1}^{\ell} \pi(i_k) \le S_\ell + \ell - p - z$, where $z = |\{j \le p \mid \pi(j) = 0\}|$.*

Thresholds. The algorithm uses the potential function to compute thresholds. Namely, at some points of the time (defined later; these points are the same for all nodes), $\mathcal{L}$ is sorted in the non-increasing order of keys. Ties are broken according to ids, which assures that the ordering computed locally is the same for all the nodes. At this point $k_1, k_2, \ldots, k_n$ denote the values of keys and they are a non-increasing sequence. The potential π is computed on the values of keys k_i and is stored in the thresholds $\varphi(i)$, i.e., we simply set $\varphi(i) := \pi(i)$ for all $i \in [1, n]$. Threshold values $\varphi(1), \ldots, \varphi(n)$ change only at those times. At any time, the packets at node i above the threshold $\varphi(i)$ 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.

Algorithm Definition. At time 0, the algorithm initializes its global variables. Namely, $\mathcal{L}$ is populated with ids of the nodes, sorted according to the values of id. 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. Sect. 1 with the description of the model).

– In round t, the processor whose position on $\mathcal{L}$ is equal to *token* transmits. It transmits a message containing a packet from its queue (if it has any) along with the size of its queue (computed after removing the transmitted packet from the queue). All nodes, including the transmitting one receive this message.

Algorithm 1. Update of global variables at time t

```
case $mode = scanning$
    if $\sum_{j=1}^{token}(k_j + p_j - \varphi_j) \leq token$ and $token < n$ then
        $token \leftarrow token + 1$
    else
        sort $\mathcal{L}$ in a non-increasing order of keys
        $\varphi \leftarrow$ the potential function of keys
        if there exists $i$ such that $k_i > \varphi_i$ then
            $token \leftarrow \min\{i \mid k_i > \varphi_i\}$
            $mode \leftarrow trimming$
        else
            $token \leftarrow 1$
case $mode = trimming$
    if $\sum_{j=1}^{n}(k_j - \varphi_j) > 0$ then
        if $k_{token} \leq \varphi_{token}$ then
            $token \leftarrow \min\{\ell > token \mid k_\ell > \varphi_\ell\}$
    else
        $token \leftarrow 1$
        $mode \leftarrow scanning$
```

– Round t is divided into three actions, executed w.l.o.g. in the following order:
  1. All nodes update key k_{token} on the list $\mathcal{L}$ as well as the value of the variable p_{token} on the basis of the message they heard in round t. Precisely, if the message from the transmitting processor contains a packet, p_i is set to 1, otherwise the round is void (i.e., the transmitting processor's queue is empty and its message contains only the information that its queue size is zero), both k_{token} and p_{token} are set to 0. That is, p_i indicates whether the node i had a nonempty queue when it held the *token* for the last time.
  2. The adversary injects an arbitrary number of packets to the system; they are appended to particular queues.
  3. Each node executes Algorithm 1. Depending on the mode, all nodes execute the instructions from the corresponding case.

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

2.1 SCAT Analysis

Below, we show that the algorithm Scat is optimal with respect to both complexity measures (maximum load and total load).

Consider an execution of algorithm Scat. 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 Scat is in trimming mode. On the other hand, the contiguous sequence of rounds in which Scat is in 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 either $\sum_{j=1}^{token}(k_j + p_j - \varphi_j) > token$ or $token = n$. If the former condition occurs, then we call such scanning cycle *balanced*.

As described previously, all packets kept by the algorithm are classified either as overhead or non-overhead packets. Intuitively, the total value of potential (i.e., the number of non-overhead packets in the system) describes the number of packets which are already "under control" 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 arbitrary number of packets in each round. However, in such case even Opt has these packets in its queues. Second, when Scat 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.[1]

Semi-potentials. By the algorithm definition, at some times 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 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 infeasible, we introduce a helper 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 for any $1 \leq i \leq n$, the following two properties hold: (i) $0 \leq \psi(i) \leq x_i$, and (ii) 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. Clearly, for sorted keys their potential function is also a semi-potential function. Moreover, it is also the "largest" possible semi-potential, as stated next.

Lemma 2. *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, $\overline{\psi} \leq \overline{\pi}$.*

[1] It can be shown 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 adversary can create an injection pattern causing the additional $\Omega(n)$ factor in the max-load competitiveness (both in multiplicative and additive components). It demonstrates the subtlety of the chosen potential function, which is essential to control scanning and trimming processes.

Changes in the Potential. 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 3. *Consider a balanced scanning cycle C in the execution of algorithm* SCAT*. 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.*

Proof. Let $k'_1, \dots, k'_n$ be the values of keys at the beginning of the cycle C. Let $k_1, \dots, k_n$ be the values of keys at the end of the cycle, before they are sorted by SCAT. Let $p_1, \dots, p_n$ be the queue non-emptiness indicators at the end of the cycle. Clearly, $k_i + p_i \geq k'_i$ for any i. As π is the potential at the beginning of the cycle, $\pi(i) \leq k'_i$ and therefore, π is the semi-potential for values $k_1 + p_1, \dots, k_n + p_n$. Thus, $\overline{\pi} \leq \sum_{i=1}^{n}(k_i + p_i)$. We show that there exists a semi-potential function ψ with respect to $k_1, \dots, k_n$, such that $\overline{\psi} = \overline{\pi} + y$. This will immediately conclude the proof as the total potential for the sorted values $k_1, \dots, k_n$ is at least $\overline{\psi}$ by Lemma 2.

Since we consider a balanced scanning cycle and the threshold values are equal to the values of π, the following two properties hold
(P1) $\sum_{j=1}^{\ell'}(k_i + p_i - \pi(i)) \leq \ell'$ for each $\ell' < \ell$, and
(P2) $\sum_{j=1}^{\ell}(k_i + p_i - \pi(i)) > \ell$,
where ℓ is the position (on the list $\mathcal{L}$) of the last node which transmits during the considered scanning cycle. We define the function:

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

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

The first property of the semi-potential function states that $0 \leq \psi(i) \leq k_i$ for all i. This condition holds trivially for $i < \ell$. For $i \geq \ell$ observe that the value of key of the i-th element on the list was also k_i at the beginning of the cycle, and thus $\pi(i) \leq k_i$. Thus, it remains to verify that $0 \leq \psi(l) \leq k_l$. As $\sum_{i=1}^{\ell} p_i$ is the number of non-void rounds of C, $\sum_{i=1}^{\ell} p_i + y = \ell$. Applying this relation to the definition of $\psi(\ell)$, we obtain that

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

Thus, using Property (P2), we obtain that $\psi(\ell) = k_i + \ell + \sum_{i=1}^{\ell}(\pi(i) - k_i - p_i) > k_\ell$. Furthermore, applying Property (P1) with $\ell' = \ell - 1$,

$$\begin{aligned} \psi(l) &= \ell + \pi(\ell) - p_\ell - \sum_{j=1}^{l-1}(k_j + p_j - \pi(j)) \\ &\geq \ell + \pi(\ell) - p_\ell - (\ell - 1) \geq \pi(\ell) \geq 0 \ . \end{aligned} \tag{1}$$

The second property of the semi-potential function states that the sum of any m values among $\psi(1), \dots, \psi(n)$ 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 $\sum_{j \in I} \psi(j) \leq S_m$.

Let y_r be the number of void rounds up to the position r, i.e., $y_r = r - \sum_{j=1}^{r} p_j$. We observe that

$$\textstyle\sum_{j=1}^{r} \psi(j) \leq \sum_{j=1}^{r} \pi(j) + y_r \ . \tag{2}$$

Indeed, if $r \geq \ell$, $y_r = y$ and the inequality follows directly from the definition of ψ, and for $r < \ell$, we use Property (P2) obtaining $\sum_{j=1}^{r} \psi(j) = \sum_{j=1}^{r} k_j \leq r + \sum_{j=1}^{r} \pi(j) - \sum_{j=1}^{r} p_j = \sum_{j=1}^{r} \pi(j) + y_r$.

Observe that if a round j is void, then $\pi(j) \leq k_j + p_j = 0$. Hence, $y_r \leq |\{j \leq r \,|\, \pi(j) = 0\}|$, and thus by Lemma 1,

$$\sum_{j \in I} \pi(j) \leq S_m + m - r - |\{j \leq r \,|\, \pi(j) = 0\}| \leq S_m + m - r - y_r \ . \tag{3}$$

Let $A_< = \{j \leq r \,|\, \psi(j) < \pi(j)\}$ and $A_> = \{j \leq r \,|\, \psi(j) > \pi(j)\}$. Observe that $\psi(j) \geq \pi(j) - 1$ for each $j \in [1, n]$ which follows from the relationship $k_j + p_j \geq \pi(j)$, the definition of ψ and (1). Therefore, $\psi(j) = \pi(j) - 1$ for any $j \in A_<$. Thus, using (2),

$$\sum_{j \in A_>} (\psi(j) - \pi(j)) = \sum_{j=1}^{r} (\psi(j) - \pi(j)) \sum_{j \in A_<} (\pi(j) - \psi(j)) \leq y_r + |A_<| \ . \tag{4}$$

Finally, combining (3) with (4), we get:

$$\begin{aligned}
\textstyle\sum_{j \in I} \psi(j) &= \textstyle\sum_{j \in I} \pi(j) + \sum_{j \in I} (\psi(j) - \pi(j)) \\
&= \textstyle\sum_{j \in I} \pi(j) + \sum_{j \in I \cap A_>} (\psi(j) - \pi(j)) - \sum_{j \in I \cap A_<} (\pi(j) - \psi(j)) \\
&\leq (S_m + m - r - y_r) + (y_r + |A_<|) - |I \cap A_<| \\
&\leq S_m + |I| + |A_<| - |I \cap A_<| - r \leq S_m \ . \qquad \square
\end{aligned}$$

Using the crucial lemma above and applying a few observations, we may compare the *current* 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 (current) number of packets exceeding the total value of threshold of the algorithm Scat by ovr_t, and the number which Opt has in queues at t by opt_t. Note that we compute these values after the adversary injects packets at time t and after the algorithm computes new values of thresholds (if it does so). Note that at time 0, all thresholds are equal to zero and since the queues of Opt and Scat are equal, i.e. $\mathrm{ovr}_0 = \mathrm{opt}_0$.

Lemma 4. *Fix any scanning or trimming cycle C starting at time t and ending at time $t + r$. Then $ovr_{t+r} - opt_{t+r}$ is at most*
1. *$ovr_t - opt_t + n$ if C is a non-balanced scanning cycle;*
2. *$ovr_t - opt_t$ if C is a balanced scanning cycle or a trimming cycle.*

Using the lemma above, by a simple induction we obtain the following result, stating that we may control the number of overhead packets.

Lemma 5. *For any cycle C starting at time t, it holds that $ovr_t \le opt_t + 2n$.*

Finally, we prove upper bounds on competitiveness of SCAT.

Proof (of Theorem 1). Fix any time $t+r$ belonging to a cycle C starting at time t. By Lemma 5, $\mathrm{ovr}_t \le \mathrm{opt}_t + 2n$. Assume that at times $t+1, t+2, \dots, t+r$, the adversary injected in total j (overhead) packets. Hence, $\mathrm{opt}_{t+r} \ge \mathrm{opt}_t + j - r$. If C is a trimming cycle, then SCAT transmits an overhead packet in each step, i.e., $\mathrm{ovr}_{t+r} = \mathrm{ovr}_t + j - r \le \mathrm{opt}_{t+r} + 2n$. If C is a scanning cycle, then its length is at most n, and thus even if SCAT does not transmit any overhead packets, then $\mathrm{ovr}_{t+r} \le \mathrm{ovr}_t + j$ and $\mathrm{opt}_{t+r} \ge \mathrm{opt}_t + j - n$. In this case, $\mathrm{ovr}_{t+r} \le \mathrm{opt}_{t+r} + 3n$.

In either case, $\mathrm{ovr}_{t+r} \le \mathrm{opt}_{t+r} + 3n$. 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 any time is at most $\mathrm{opt}_{t+r} + 3n + 2n = n \cdot (\mathrm{opt}_{t+r}/n) + 5n$. As the maximum load of OPT at step $t+r$ is at least opt_{t+r}/n, the second part of the theorem follows. □

3 Stochastic Model

In this section we assume that packets are injected according to a random distribution, defined by the sequence of numbers $p_1, \dots, p_n \in (0,1)$. In each step, for each queue independently, one packet is injected into the queue j with probability p_j and no packet is injected with probability $1 - p_j$. Our goal is to analyze the total load of *deterministic* distributed algorithms in such scenario.

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 step, 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 $S_0, S_1, S_2, \dots$ are non-negative integers corresponding to the total load at consecutive times. In particular, S_0 is the initial number of packets in all buffers. Let Y_t be the random variable denoting the number of packets injected in step t. Recall that by the definition of our process, all Y_t are identically and independently distributed, their support is the set $\{0, \dots, n\}$ and their mean is equal to $\sum_{j=1}^{n} p_j$. The transitions between consecutive states is then defined by

$$S_t = \begin{cases} S_{t-1} + Y_t - 1 & \text{if } S_{t-1} > 0 \\ S_{t-1} + \max\{Y_t - 1, 0\} & \text{if } S_{t-1} = 0 \end{cases}$$

In the following, we restrict our attention to time-homogeneous Markov chains only. For any such 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 $E_C(S \to S')$. We are interested in the event of Opt reaching the empty queues state, and therefore we concentrate on bounding the terms $P_C(S_0 \to 0)$ and $E_{\text{Opt}}(S_0 \to 0)$. Note that the finiteness of $E_{\text{Opt}}(S_0 \to 0)$ trivially implies that $P_{\text{Opt}}(S_0 \to 0) = 1$. The goal of this section is to present tight conditions on $\sum_{j=1}^{n} p_j$ which assure that $P_{\text{Opt}}(S_0 \to 0) = 1$ or $E_{\text{Opt}}(S_0 \to 0)$ is finite.

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. Furthermore, as in each step there is a positive probability that the number of packets remains the same (i.e., the state does not change), Opt is aperiodic.

Lemma 6. *Fix any starting state S_0. If $\sum_{j=1}^{n} p_j < 1$, then $E_{\text{Opt}}(S_0 \to 0)$ is finite.*

Lemma 7. *Fix any starting state S_0. If $\sum_{j=1}^{n} p_j = 1$, then $P_{\text{Opt}}(S_0 \to 0) = 1$*

By combining Lemmas 6 and 7, we immediately get the following corollary.

Corollary 1. *Fix any starting state S_0. It holds that $P_{\text{Opt}}(S_0 \to 0) = 1$, provided $\sum_{j=1}^{n} p_j \leq 1$. Furthermore, if $\sum_{j=1}^{n} p_j < 1$, then $E_{\text{Opt}}(S_0 \to 0)$ is finite.*

Distributed Solution. Now, we move our attention to on-line deterministic algorithms. Since an online algorithm is not able to achieve better performance than Opt, the properties of Opt motivate the following definition.

Definition 1. *An algorithm* Alg *is stochastically optimal when both conditions hold:*
(i) $P_{\text{Alg}}(S \to 0) = 1$ for any $S \geq 0$ if $\sum_{i=1}^{n} p_i \leq 1$,
(ii) $E_{\text{Alg}}(S \to 0)$ is finite for any $S \geq 0$ if $\sum_{i=1}^{n} p_i < 1$.

Lemma 8. *Fix any monotonic functions $f, g : \mathbb{N} \to \mathbb{N}$. Assume that a deterministic distributed on-line algorithm* Alg *is $(1, f(n))$-competitive with respect to total load. Assume that given m packets in its queues,* Alg *transmits them all in the next $g(m)$ steps, provided no packet is injected in this period. Then,* Alg *is stochastically optimal.*

Corollary 2. Scat *is stochastically optimal.*

Proof. It is sufficient to fix the functions f and g satisfying the conditions of Lemma 8. By Theorem 1, $f(n) = O(n^2)$. Now assume that Scat 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. □

4 Conclusions 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 solutions. We also show a transformation from the world of competitive analysis of distributed online queuing into distributed stochastic queuing. An interesting open problem is to analyze other measures of performance, e.g., related to timing or energy efficiency, in similar online distributed frameworks.

A Remark on Message Size. In the algorithm Scat, a node holding a token attaches the current number of packets in its queue to the transmitted packet. In order to reduce the number of auxiliary bits to $O(\log n)$, the minimum of $2n$ and the current size of the queue may be sent by the node holding a token.

References

1. Anantharamu, L., Chlebus, B.S., Kowalski, D.R., Rokicki, M.A.: Deterministic broadcast on multiple access channels. In: INFOCOM, pp. 146–150. IEEE (2010)
2. Andrews, M., Awerbuch, B., Fernández, A., Leighton, F.T., Liu, Z., Kleinberg, J.M.: Universal-stability results and performance bounds for greedy contention-resolution protocols. J. ACM 48(1), 39–69 (2001)
3. Bar-Noy, A., Freund, A., Landa, S., Naor, J.S.: Competitive on-line switching policies. In: ACM-SIAM SODA, pp. 525–534 (2002)
4. Bianchi, G.: Performance analysis of the IEEE 802.11 distributed coordination function. IEEE Journal on Selected Areas in Communications 18, 535–547 (2000)
5. Borodin, A., El-Yaniv, R.: Online Computation and Competitive Analysis. Cambridge University Press (1998)
6. Borodin, A., Kleinberg, J.M., Raghavan, P., Sudan, M., Williamson, D.P.: Adversarial queuing theory. J. ACM 48(1), 13–38 (2001)
7. Chlebus, B.: Randomized communication in radio networks. In: Pardalos, P.M., Rajasekaran, S., Reif, J.H., Rolim, J.D.P. (eds.) Handbook on Randomized Computing, vol. I, pp. 401–456. Kluwer Academic (2001)
8. Chlebus, B.S., Kowalski, D.R., Rokicki, M.A.: Maximum throughput of multiple access channels in adversarial environments. Distributed Comp. 22(2), 93–116 (2009)
9. Chung, K.L., Fuchs, W.: On the distribution of values of sums of random variables. Memoirs of the AMS 6, 1–12 (1951)
10. Fleischer, R., Koga, H.: Balanced scheduling toward loss-free packet queuing and delay fairness. Algorithmica 38, 363–376 (2004)
11. Foster, F.G.: On the stochastic matrices associated with certain queueing processes. Ann. Math Statist. 24, 355–360 (1953)
12. Gallager, R.G.: A perspective on multiaccess channels. IEEE Transactions on Information Theory 31(2), 124–142 (1985)
13. Håstad, J., Leighton, F.T., Rogoff, B.: Analysis of backoff protocols for multiple access channels. SIAM J. Comput. 25(4), 740–774 (1996)
14. Richa, A.W., Scheideler, C., Schmid, S., Zhang, J.: Competitive and fair medium access despite reactive jamming. In: ICDCS, pp. 507–516 (2011)