Dynamic sharing of a multiple access channel

Marcin Bienkowski, Marek Klonowski, Miroslaw Korzeniowski, Dariusz R. Kowalski

Symposium on Theoretical Aspects of Computer Science 2010 (Nancy, France), pp. 83-94
www.stacs-conf.org

DYNAMIC SHARING OF A MULTIPLE ACCESS CHANNEL

MARCIN BIENKOWSKI [1] AND MAREK KLONOWSKI [2] AND MIROSLAW KORZENIOWSKI [2] AND DARIUSZ R. KOWALSKI [3]

[1] Institute of Computer Science, University of Wrocław, Poland

[2] Institute of Mathematics and Computer Science, Wrocław University of Technology, Poland

[3] Department of Computer Science, University of Liverpool, UK

In this paper we consider the mutual exclusion problem on a multiple access channel. Mutual exclusion is one of the fundamental problems in distributed computing. In the classic version of this problem, n processes perform a concurrent program which occasionally triggers some of them to use shared resources, such as memory, communication channel, device, etc. The goal is to design a distributed algorithm to control entries and exits to/from the shared resource in such a way that in any time there is at most one process accessing it. We consider both the classic and a slightly weaker version of mutual exclusion, called ε-mutual-exclusion, where for each period of a process staying in the critical section the probability that there is some other process in the critical section is at most ε. We show that there are channel settings, where the classic mutual exclusion is not feasible even for randomized algorithms, while ε-mutual-exclusion is. In more relaxed channel settings, we prove an exponential gap between the makespan complexity of the classic mutual exclusion problem and its weaker ε-exclusion version. We also show how to guarantee fairness of mutual exclusion algorithms, i.e., that each process that wants to enter the critical section will eventually succeed.

1. Introduction

In this paper we consider randomized algorithms for mutual exclusion: one of the fundamental problems in distributed computing. We assume that there are n different processes labeled from 0 to $n-1$ communicating through a multiple access channel (MAC). The computation and communication proceed in synchronous slots, also called *rounds*. In the mutual exclusion problem, each process performs a concurrent program and occasionally requires exclusive access to shared resources. The part of the code corresponding to this exclusive access is called a *critical section*. The goal is to provide a mechanism that controls

1998 ACM Subject Classification: C.1.4 Parallel Architectures, C.2.1 Network Architecture and Design, F.2.2 Nonnumerical Algorithms and Problems. Supported by Polish Ministry of Science and Higher Education grants no N N206 2573 35 and N N206 1723 33, and by the Engineering and Physical Sciences Research Council [grant number EP/G023018/1].

Key words and phrases: distributed algorithms, multiple access channel, mutual exclusion.

entering and exiting the critical section and guarantees exclusive access at any time. The main challenge is that the designed mechanism must be universal, in the sense that exclusive access must be guaranteed regardless of the times of access requests made by other processes.

Multiple Access Channel (MAC). We consider a multiple access channel as both communication medium and the shared-access device. As a communication medium, MAC allows each process either to transmit or listen to the channel at a round,[1] and moreover, if more than one process transmits, then a *collision* (signal interference) takes place. Depending on the devices used in the system, there are several additional settings of MAC that need to be considered. One of them is the ability of a process to distinguish between background noise when no process transmits (also called *silence*) and collision. If such capability is present at each process, we call the model *with collision detection* (CD for short); if no process has such ability, then we call the setting *without collision detection* (no-CD). Another feature of the model is a constant access to the global clock (GC for short) by all processes or no such access by any of them (no-GC). The third parameter to be considered is a knowledge of the total number of available processes n (KN for short) or the lack of it (no-KN).

Mutual Exclusion Problem. In this problem, each concurrent process executes a protocol partitioned into the following four sections:

Entry: the part of the protocol executed in preparation for entering the critical section;
Critical: the part of the protocol to be protected from concurrent execution;
Exit: the part of the protocol executed on leaving the critical section;
Remainder: the rest of the protocol.

These sections are executed cyclically in the order: *remainder, entry, critical, and exit.* Intuitively, the remainder section corresponds to local computation of a process, and the critical section corresponds to the access to the shared object (the channel in our case); though the particular purpose and operations done within each of these sections are not a part of the problem. Sections entry and exit are the parts that control switching between remainder and critical sections in a process, in order to assure some desired properties of the whole system.

In the traditional mutual exclusion problem, as defined in [1, 16] in the context of shared-memory model, the adversary controls the sections remainder and critical. In particular, she controls their duration in each cycle, subject only to the obvious assumptions that this duration in each cycle is finite or the last performed section is the remainder one. The mutual exclusion algorithm, on the other hand, provides a protocol for the entry and exit sections of each process. In this sense, the mutual exclusion problem can be seen as a game between the adversary controlling the lengths of remainder and critical sections of each process (each such section for each process may have different length) and the algorithm controlling entry and exit sections. The goal of the algorithm is to guarantee several useful properties of the execution (to be defined later), while the goal of the adversary is to prevent it. Note that the sections controlled by the adversary and those controlled by the algorithm are interleaved in the execution. Additionally, in order to make the game fair, it is typically

[1]Most of the previous work on MAC, motivated by Ethernet applications, assumed that a process can transmit and listen simultaneously; our work instead follows the recent trends of wireless applications where such simultaneous activities are excluded due to physical constraints.

assumed that every variable used by the algorithm, i.e., in the entry and exit sections, cannot be modified by the adversary in the critical and remainder sections, and vice versa, i.e., no variables used by the adversary in the remainder and critical sections can be accessed by the algorithm.

In the model of communication over MAC, a process in the entry or the exit section can do the following in a single round: perform some action on the channel (either transmit a message or listen), do some local computation, and change its section either from entry to critical or from exit to remainder. We assume that changing sections occurs momentarily between consecutive rounds, i.e., in each round a process is exactly in one section of the protocol.

Since a multiple-access channel is both the only communication medium and the exclusively shared object, additional constraints, different from the classic ones regarding e.g., shared memory objects, must be imposed:

- no process in the remainder section is allowed to transmit on the channel;
- a process in the critical section has to transmit a message on the channel each round until it moves to the exit section, and each such message must be labelled *critical*; we call them *critical messages*.

If any of these conditions was violated, the adversary would have an unlimited power of creating collisions on the channel, and thus preventing any communication.

A classic mutual exclusion algorithm should satisfy the following three properties for any round i of its execution:

Exclusion: at most one process is in the *critical* section in round i.

Unobstructed exit: if a process p is in the exit section at round i, then process p will switch to the remainder section eventually after round i.

No deadlock: if there is a process in the entry section at round i, then *some* process will enter the critical section eventually after round i.

To strengthen the quality of service guaranteed by mutual exclusion algorithms, the following property — stronger than no-deadlock — has been considered:

No lockout: if a process p is in the entry section at round i, then *process* p will enter the critical section eventually after round i.

Note that — to some extent — this property ensures fairness: each process demanding an access to the critical section will eventually get it.

As we show, in some cases the exclusion condition is impossible or very costly to achieve. Therefore, we also consider a slightly weaker condition:

ε-exclusion: for every process p and for every time interval in which p is continuously in the critical section, the probability that in any round of this time interval there is another process being in the critical section is at most ε.

Intuitively, ε-exclusion guarantees mutual exclusion "locally", i.e., for every single execution of the critical section by a process, with probability at least $1-\varepsilon$. The version of the problem satisfying ε-exclusion condition is called *ε-mutual-exclusion*.

Complexity Measure. We use the *makespan* measure, as defined in [8] in the context of deterministic algorithms. Makespan of an execution of a given deterministic mutual exclusion algorithm is defined as the maximum length of a time interval in which there is some process in the entry section and there is no process in the critical section. Taking maximum of such values over all possible executions defines the makespan of the algorithm.

In order to define expected makespan, suitable for randomized algorithms considered in this work, we need more formal definitions of an adversarial strategy. Let $\mathcal{P}$ be a strategy of the adversary, defined as a set of n sequences, where each sequence corresponds to a different process and contains, subsequently interleaved, lengths of remainder and critical sections of the corresponding process. We assume that each sequence is either infinite or of even length; the latter condition means that after the last critical section the corresponding process runs the remainder section forever. For a given mutual exclusion algorithm ALG and adversarial strategy $\mathcal{P}$, we define $L(\text{ALG}, \mathcal{P})$ as a random variable equal to the maximum length of a time interval in which there is some process in the entry section and there is no process in the critical section in an execution of ALG run against fixed strategy $\mathcal{P}$. The expected makespan of algorithm ALG is defined as the maximum of expected values of $L(\text{ALG}, \mathcal{P})$, taken over all adversarial strategies $\mathcal{P}$. Note that every algorithm with makespan bounded for all executions satisfies no-deadlock property, but not necessarily no-lockout.

For the ε-mutual-exclusion problem, defining makespan is a bit more subtle. We call an execution *admissible* if the mutual exclusion property is always fulfilled, i.e., no two processes are in the critical section in the same round. Then in the computation of the (expected) makespan, we neglect non-admissible executions.

1.1. Our Results

We consider the mutual exclusion problem and its weaker ε-exclusion version in the multiple access channel. Unlike the previous paper [8], where only no-deadlock property was guaranteed, we also focus on fairness. Also in contrast to the previous work on mutual exclusion on MAC, we mostly study randomized solutions. In the case of the mutual exclusion problem, we allow randomized algorithms to have variable execution time but they have to be always correct. On the other hand, a randomized solution for the ε-mutual-exclusion problem is allowed to err with some small probability ε. Thus, for the former problem, we require Las Vegas type of solution, whereas for the latter we admit Monte Carlo algorithms. Note that very small (e.g., comparable with probability of hardware failure) risk of failure (i.e., situation wherein two or more processes are in the critical section at the same round) is negligible from a practical point of view.

We show that for the most severe channel setting, i.e., no-CD, no-GC and no-KN, mutual exclusion is not feasible even for randomized algorithms (cf. Section 2).

In a more relaxed setting, there is an exponential gap between the complexity of the mutual exclusion problem and the ε-mutual-exclusion problem. Concretely, we prove that the expected makespan of (randomized) solutions for the mutual exclusion problem in the no-CD setting is $\Omega(n)$, even if the algorithm knows n, has access to the global clock (cf. Section 2), and even if only no-deadlock property is required. On the other hand, for the ε-mutual-exclusion problem, we construct a randomized algorithm, requiring only the knowledge of n, which guarantees no-lockout property, and whose makespan is $O(\log n \cdot \log(1/\varepsilon))$ (cf. Sections 3.2 and 4).

When collision detection is available and only no-deadlock property is required, we show that the makespan of any mutual exclusion algorithm is at least $\Omega(\log n)$ (cf. Section 2) and we construct an algorithm for the ε-mutual-exclusion problem with expected makespan $O(\log\log n + \log(1/\varepsilon))$ (cf. Section 3.3). Further, we show how to modify this algorithm to guarantee no-lockout property as well; its expected makespan becomes then $O(\log n + \log(1/\varepsilon))$ (cf. Sections 3.3 and 4).

Finally, if we do not require no-lockout property, we show how to solve the ε-mutual-exclusion problem in makespan $O(\log n \cdot \log(1/\varepsilon))$, where only the global clock is available (cf. Section 3.1).

We also present a generic method that, taking a mutual exclusion algorithm with no-deadlock property, turns it into the one satisfying stronger no-lockout condition. This method applied to the deterministic algorithms from [8] produces efficient deterministic solutions satisfying the no-lockout property.

Due to space limitations, the missing details and proofs will appear in the full version of the paper.

1.2. Previous and Related Work

The multiple access channel is a well-studied model of communication. In many problems considered in this setting, one of the most important issues is to assure that successful transmissions occur in the computation. These problems are often called *selection problems*. They differ from the mutual exclusion problem by the fact that they focus on successful transmissions within a bounded length period, while mutual exclusion provides control mechanism for dynamic and possibly unbounded computation. In particular, it includes recovering from long periods of cumulative requests for the critical section as well as from long periods containing no request. Additionally, selection problems were considered typically in the context of Ethernet or combinatorial group testing, and as such they allowed a process to transmit and to listen simultaneously, which is not the case in our model motivated by wireless applications. Selection problems can be further split into two categories. In the static selection problems, it is assumed that a subset of processes become active at the same time and a subset of them must eventually transmit successfully. Several scenarios and model settings, including parameters considered in this work such as CD/no-CD, GC/no-GC, KN/no-KN, randomization/determinism, were considered in this context, see e.g., [2, 4, 7, 11, 12, 14, 15, 17, 18, 19]. In the *wake-up* problem, processes are awaken in (possibly) different rounds and the goal is to assure that there will be a round with successful transmission ("awakening" the whole channel) shortly after the first process is awaken, see, e.g., [5, 9, 13].

More dynamic kinds of problems, such as transmission of dynamically arriving packets, were also considered in the context of MAC. In the (dynamic) packet transmission problem, the aim is to obtain bounded throughput and bounded latency. Two models of packet arrival were considered: stochastic (cf., [10]) and adversarial queuing (cf., [3, 6]). There are two substantial differences between these settings and our work. First, the adversaries imposing dynamic packet arrival are different than the adversary simulating execution of concurrent protocol. Second, as already mentioned in the context of selection problems, these papers were inspired by Ethernet applications where it is typically allowed to transmit and listen simultaneously.

In a very recent paper [8] *deterministic* algorithms for mutual exclusion problem in MAC under different settings (CD, GC, KN) were studied. The authors proved that with none of those three characteristics mutual exclusion is infeasible. Moreover, they presented an optimal — in terms of the makespan measure — $O(\log n)$ round algorithm for the model with CD. They also developed algorithms achieving makespan $O(n \log^2 n)$ in the models with GC or KN only, which, in view of the lower bound $\Omega(n)$ on deterministic solutions proved for any model with no-CD, is close to optimal. Our paper differs from [8] in three

ways. First, we consider both deterministic and randomized solutions. Second, for the sake of efficiency we introduce the ε-mutual-exclusion problem. Third, we study fairness of protocols, which means that we consider also no-lockout property.

2. Lower Bounds for the Mutual Exclusion Problem

In our lower bounds, we use the concept of transmission schedules to capture transmission/listening activity of processes in the entry or exit section. Transmission schedule of a process p can be regarded as a binary sequence π_p describing the subsequent communication actions of the process. The sequence can be finite or infinite. For non-negative integer i, $\pi_p(i) = 1$ means that process p transmits in round i after starting its current section, while $\pi_p(i) = 0$ means that the process listens in round i. We assume that round 0 is the round in which the process starts its current run of the entry or the exit section.

The following results extend the lower bounds and impossibility results for deterministic mutual exclusion proved in [8] to randomized solutions. All the presented lower bounds work even if we do not require no-lockout, but a weaker no-deadlock property.

Theorem 2.1. *There is no randomized mutual exclusion algorithm with no-deadlock property holding with a positive probability in the setting without collision detection, without global clock and without knowledge of the number n of processes.*

Theorem 2.2. *The expected makespan of any randomized mutual exclusion algorithm is at least $\log n$, even in the setting with collision detection, with global clock and with knowledge of the number n of processes.*

Theorem 2.3. *The expected makespan of any randomized mutual exclusion algorithm is at least $n/2$ in the absence of collision detection capability, even in the setting with global clock and with knowledge of the number n of processes.*

Proof. To arrive at a contradiction, let $\mathcal{R}$ be a randomized mutual exclusion algorithm, whose expected makespan is c, where $c < n/2$. We show that there exists an execution violating mutual exclusion.

Let $\mathcal{E}_p^*$, for process p, be the set of all possible executions of the first entry section of algorithm $\mathcal{R}$ by process p under the assumption that it starts its first entry section in the global round 1 and there is no other process starting within the first $n/2$ rounds. Note that during each execution in $\mathcal{E}_p^*$ process p hears only noise (i.e., silence or collision, which are indistinguishable due to the lack of collision detection) from the channel when listening. Observe also that the optimum algorithm needs only one round to let process p enter the critical section under the considered adversarial scenario. Therefore, by the probabilistic method, there is an execution $\mathcal{E}_p$ in set $\mathcal{E}_p^*$ where process p enters the critical section within the first $n/2 - 1$ rounds. Let π_p be the transmission schedule of process p during $\mathcal{E}_p$.

Consider all sequences π_p over all processes $0 \leq p < n$. We construct execution $\mathcal{E}$ contradicting mutual exclusion as follows. First, we need to select a set of processes that start their first entry sections in round 1, while the others stay in the remainder section till at least round $n/2$. Let $P_0 = \{0, \ldots, n-1\}$. For every non-negative integer j, we define recursively

$$\begin{aligned} P_{2j+1} &= P_{2j} \setminus \left\{ p \in P_{2j} : \exists_{i \in [1, n/2-1]} \left(\pi_p(i) = 1 \ \& \ \forall_{q \in P_{2j}, q \neq p} \ \pi_q(i) = 0 \right) \right\} \ , \\ P_{2j+2} &= P_{2j+1} \setminus \left\{ p \in P_{2j+1} : \exists_{i \in [1, n/2-1]} \left(|\pi_p| = i \ \& \ \forall_{q \in P_{2j+1}} \ |\pi_q| > i \right) \right\} \ . \end{aligned}$$

Intuitively, set P_{2j+1} is obtained from P_{2j} by removing processes p that could be single transmitters in some round in the interval $[1, n/2-1]$ while transmitting according to their schedules π_p. Set P_{2j+2} is constructed by removing a process with the shortest transmission schedule, if there is only one such process. Observe that sequence $\{P_j\}_{j\geq 0}$ is bounded and monotonically non-increasing (in the sense of set inclusion), therefore it stabilizes on some set P^*. Observe that

(1) $|P^*| \geq 2$, since for each round $i \in [1, n/2-1]$ there is at most one process removed from some set P_{2j} while constructing the consecutive set P_{2j+1} (after such removal no remaining process has 1 in position i of its schedule) and at most one process removed from some set $P_{2j'+1}$ while constructing the consecutive set $P_{2j'+2}$ (after such removal no remaining process p finishes its transmission schedule π_p in round i); as there are $n/2-1$ considered rounds, at most $n-2$ processes can be removed throughout the construction;

(2) there is no round $i \in [1, n/2-1]$ such that there is only one process $p \in P^*$ satisfying $\pi_p(i) = 1$; this follows from the fact that P^* is a fixed point of the sequence $\{P_j\}_{j\geq 0}$, i.e., it does not change while applying the odd-step rule of the construction;

(3) there are at least two processes $p, q \in P^*$ with the shortest transmission schedules π_p, π_q, i.e., $|\pi_p| = |\pi_q|$ and for every process $r \in P^*$, $|\pi_r| \geq |\pi_p|$; this again follows from the fact that P^* is a fixed point of the sequence $\{P_j\}_{j\geq 0}$, i.e., it does not change while applying the even-step rule of the construction.

Having subset P^* of processes, the adversary starts first entry sections for all processes in P^* in the very first round, while she delays others (they remain in the remainder section) by round $n/2$. Note that before round 1 of the constructed execution $\mathcal{E}$, a process $p \in P^*$ cannot distinguish $\mathcal{E}$ from $\mathcal{E}_p$, therefore it may decide to do the same as in $\mathcal{E}_p$, i.e., to set its first position of transmission schedule to $\pi_p(1)$. If this happens for all processes in P^*, by the second property of this set there is no single transmitter in round 1, and therefore all listening processes hear the noise (recall that silence is not distinguishable from collision in the considered setting). This construction and the output of the first round can be inductively extended up to round $|\pi_p|$, where $p \in P^*$ is a process with the shortest schedule π_p among processes in P^*. This is because from the point of view of a process $q \in P^*$ the previously constructed prefix of $\mathcal{E}$ is not distinguishable from the corresponding prefix of execution $\mathcal{E}_q$; indeed, the transmission schedules are the same and the feedback from the channel is silence whenever the process listens. Finally, by the very same reason, at the end of round $|\pi_p|$ all processes $q \in P^*$ with $|\pi_q| = |\pi_p|$ are allowed to do in $\mathcal{E}$ the same action as in $\mathcal{E}_q$, that is, to enter the critical section. By the third property of set P^*, there is at least one such process $q \in P^*$ different than p. This violates the exclusion property that should hold for the constructed execution $\mathcal{E}$. ∎

3. Algorithms for the ε-Mutual-Exclusion Problem

In this section, we present randomized algorithms solving the ε-mutual-exclusion problem for various scenarios, differing in the channel capabilities (e.g., CD/no-CD, KN/no-KN, GC/no-GC). The algorithms presented in this section, work solely in entry sections, i.e., their exit sections are empty; these algorithms guarantee only no-deadlock property. However, in Section 4, we show how to add exit section subroutines to most of our algorithms to guarantee the no-lockout property while keeping bounded makespan. In our algorithms,

we extend some techniques developed in the context of other related problems, such as the wake-up problem [13] and the leader election problem [19].

Throughout this section, we use the following notation. We say that there is a *successful transmission* in a given round if in this round one process transmits and other processes do not transmit. By saying that a process *resigns*, we mean that it will not try to enter the critical section and will not attempt to transmit anything until another process starts the exit section.

3.1. Only Global Clock Available

In the model with global clock, we modify the *Increase_From_Square* algorithm [13], which solves the wake-up problem. The purpose of our modification is to assure the stopping property. This is a nontrivial task in a scenario without collision detection and this property was not present in the original wake-up algorithm. Intuitively, after one process successfully transmits, it should enter the critical section. However, first of all it might not be aware that it succeeded. Second, between a successful transmission and entering the critical section, some other processes may start their entry sections. The details will be presented in the full version of this paper.

Theorem 3.1. *There is an ε-mutual-exclusion algorithm, using a modified algorithm* Increase_From_Square *as a subroutine for the entry section, with makespan $O(\log n \cdot \log(1/\varepsilon))$ in the model without global clock.*

3.2. Only Number of Processes Known

In this scenario, we build our solution based on the *Probability_Increase* algorithm of [13]. In this algorithm, each process works in $\Theta(\log n)$ phases, each lasting $\Theta(\log(1/\varepsilon))$ rounds. In each round of phase i, a process transmits with probability 2^{-i}.

Lemma 3.2 ([13])**.** *If all processes use the algorithm* Probability_Increase *after being awaken, then there is a successful transmission in time $k = O(\log n \cdot \log(1/\varepsilon))$ with probability at least $1 - \varepsilon$.*

We describe how to modify the *Probability_Increase* algorithm to meet the requirements of ε-exclusion. When a process enters the entry section, it first switches to the listening mode and stays in this mode for $k = O(\log n \cdot \log(1/\varepsilon))$ rounds. If within this time it hears another process, it resigns. Afterwards, the process starts to execute the *Probability_Increase* algorithm. Whenever it is not transmitting, it listens, and when it hears a message from another process, it resigns. After executing k rounds of the listening mode and the following k rounds of *Probability_Increase* without resigning, the process enters the critical section. Using this algorithm, the following result can be proved.

Theorem 3.3. *There is an ε-mutual-exclusion algorithm, using a modified algorithm* Probability_Increase *as a subroutine for the entry section, with makespan $O(\log n \cdot \log(1/\varepsilon))$ in the KN model.*

Proof. Let k be as defined above in the algorithm definition. Let t be a round in the execution in which there is at least one process in the entry section, no process in the exit or critical section, and such that there was no process in the entry section in the previous round $t - 1$. Let P be the set of processes which are in their entry sections at round $t + k$.

First, we observe that processes which enter their entry section in round $t+k+1$ or later, i.e., all processes that are not in set P, do not transmit in the time period $[t, t+2k]$. By Lemma 3.2, with probability $1-\varepsilon$, there is a process in P which successfully transmits at some round in $[t+k, t+2k)$. Let $t+k \leq r < t+2k$ be the first such round, and $p \in P$ be the process transmitting successfully in round r. Note that all other processes being in the entry section resign at this round, and all processes that start their entry sections after round r do not transmit by round $r+k$. Therefore, p does not hear anything before it finishes its *Probability_Increase* subroutine (in the next at most $k-1$ rounds after r), which implies that it enters the critical section by round $r+k-1 < t+2k$. ∎

3.3. Only Collision Detection Available

In this scenario, the main idea behind our algorithm is as follows. First, we show how to solve a *static case* of the ε-mutual-exclusion problem, i.e., the case where there is a subset S of processes which start their entry sections at round 1 and no process is activated later. Later, we show that we are then able to solve ε-mutual-exclusion problem in (asymptotically) the same time. In what follows, we assume that whenever a process does not transmit, it listens.

To solve the static case, we first run a simple *Check_If_Single* subroutine, which, with probability at least $1-\varepsilon$, determines whether there is one active processes or more. In the former case, this process may simply enter the critical section. In the latter, we simulate Willard's algorithm [19], which works in expected time $\log\log n + o(\log\log n)$. The simulation is required, as the original algorithm of [19] assumes that each process can simultaneously transmit and listen in each round. The idea of this simulation is that for each message sent, all listening processes acknowledge it in the next round.

Lemma 3.4. *If there are at least two active processes, it is possible to simulate one round taken in the model in which a process may simultaneously transmit and listen, in two rounds of our model in the setting with collision detection.*

As mentioned above, another building block is a procedure *Check_If_Single*. The algorithm assumes that there is a set of processes which start this procedure simultaneously. The procedure consists of $2\cdot\log(1/\varepsilon)$ rounds. In each odd round, process i tosses a symmetric coin, i.e., with probability $1/2$ of success, to choose whether it transmits in the current round and listens in the next round, or vice versa. If the process never hears anything, it enters the critical section at the end of the procedure.

Lemma 3.5. *Assume k processes execute the procedure* Check_If_Single. *If $k=1$, then the only process enters the critical section. If $k \geq 2$, then with probability $1-\varepsilon$, no process enters the critical section.*

Proof. The first claim holds trivially. For showing the second one, we fix an odd-even pair of rounds. Let E denote the event that there is a process, which does not hear anything in this pair of rounds. For this to happen all processes running *Check_If_Single* have to transmit in the odd round or all have to transmit in the even round. Thus, $\Pr[E] = 2\cdot 1/2^k = 1/2^{k-1} \leq 1/2$. Since the transmissions in different pairs of rounds are independent, the probability that there exists a process which does not hear anything during the whole algorithm, and thus enters the critical section, is at most $(1/2)^{\log(1/\varepsilon)} = \varepsilon$. ∎

We may now describe an algorithm solving the static ε-mutual-exclusion problem. Let S be a subset of processes which simultaneously start their entry sections. In the first $2\log(1/\varepsilon)$ rounds, the processes execute the procedure *Check_If_Single*. Then the processes that did not enter the critical section, run a simulation of Willard's algorithm, as described in Lemma 3.4. The processes that transmit successfully, enter the critical section. Using this algorithm, the following result can be proved.

Theorem 3.6. *In the scenario with collision detection, there is an algorithm solving the static ε-mutual-exclusion problem with expected makespan $O(\log\log n + \log(1/\varepsilon))$.*

Proof. Consider the algorithm described above, based on the procedure *Check_If_Single*. If there is only one process starting its entry section, it enters the critical section right after the procedure *Check_If_Single* (which takes $O(\log(1/\varepsilon))$ rounds). If there is more than one process, with probability $1-\varepsilon$ they do not enter the critical section after this procedure and they all simultaneously start the simulation of Willard's algorithm. By the property of Willard's algorithm [19] and by Lemma 3.4, in expectation there is a successful transmission in $O(\log\log n)$ rounds. ∎

It remains to show that we may use an algorithm for static version of ε-mutual-exclusion to solve the general version of the ε-mutual-exclusion problem. The idea is to synchronize processes at the beginning, and then to transmit a "busy" signal in every second round. New processes starting their entry section note this signal and will not compete for the critical section, until an exit section releases the shared channel.

Theorem 3.7. *If there exists an algorithm* Alg *for the static ε-mutual-exclusion problem with (expected) makespan T in the model with collision detection, then there exists an algorithm* Alg$'$ *for the ε-mutual-exclusion problem with (expected) makespan $2+2\cdot T$ in the same setting.*

4. Fairness

The algorithms shown in [8] and Section 3 do not consider the no-lockout property, i.e., it may happen that a process never gets out of its entry section, as other processes exchange access to the critical section among themselves. We show how to modify algorithms satisfying no-deadlock property (in particular, the algorithms from [8]), so that the no-lockout property is fulfilled. Moreover, our transformation allows to express the (expected) makespan of obtained fair protocols in terms of the (expected) makespan of the original weaker protocols.

Each process maintains an additional local counter of losses. When it starts its entry section, it sets its counter to zero and whenever it loses the competition for the critical section, i.e., when some other process enters the critical section, it increases this counter by one. When a process enters its exit section, it becomes a guard: it helps processes currently being in the entry section to choose one of them with the highest loss counter. How high the loss counter can grow is bounded by the number of processes in their entry sections at the moment when the considered process entered its current entry section. Thus, also the time after which the process will enter the critical section is bounded.

Lemma 4.1. *If either collision detection is available or the number of nodes is known, it is possible to transform a mutual exclusion algorithm with (expected) makespan T into*

an algorithm, which also guarantees the no-lockout property and has (expected) makespan $O(T + \log n)$.

Proof. Here we only describe a transformation for the CD scenario; the analysis and the variant for KN will appear in the full version of the paper. Let ALG be a given subroutine for the entry section, satisfying no-deadlock property. In order to use it for an entry procedure satisfying stronger no-lockout property, we slow down algorithm ALG three times, by preceding each original round by two additional rounds: in the first one the process transmits signal 1, while in the second one it only listens. We call the obtained subroutine ALG′.

We also need the following selection subroutine. Assume there is a single guard and a subset (may be empty) of other processes, called *competing processes*. They all start the selection subroutine in the same round. The goal is to elect one of the competing processes to enter the critical section. The subroutine is partitioned into *blocks*, each consisting of three rounds. In the first two rounds of each block only the guard transmits, and the signals are 1 and 0, respectively. The purpose of these rounds is to assure that processes that start their entry sections later will not disturb the selection subroutine. The competition, which is essentially a binary search for the highest loss counter of the competing processes, proceeds in the following phases. In the ith block of the first phase all processes whose loss counter is at least 2^i broadcast a 0 (after the guard's 10), all other processes listen. The phase ends with a block i when silence is heard, thus all competitors and the guard know that the highest loss counter is between 2^{i-1} and 2^i. Then a binary search is performed in additional $O(i)$ blocks in similar manner. Additional binary search is performed to choose one process (the one with the minimum id) from all processes with the same maximal number of losses.

We now describe a procedure governing the exit section. Recall that a process being in the critical section always broadcasts the critical message to let others know that the channel is occupied. For the purpose of this reduction and its analysis, we denote the critical message by a single bit 1 (this is only technical assumption to simplify the proof arguments). When the process starts its exit section and becomes a guard, it transmits a 0 in the first round and listens in the second round. If the guard hears silence then it switches to the remainder section; otherwise it participates in the selection subroutine described above.

Each process starting its entry section listens for three rounds. If it hears silence during all these rounds, it starts executing ALG′ until some process enters the critical section (it is guaranteed by no-deadlock property of ALG, and can be extended to ALG′ as well); then it resets its state and starts again its entry section procedure with round counter 1. It also resets its state and starts again with round counter 1 in case it hears anything different from $1, 1, 1$ and $1, 1, 0$ during the first three rounds of listening. In the remaining third case, i.e., when the process has heard $1, 1, 1$ or $1, 1, 0$, it keeps listening until the first round t, counting from the first listening round in this run, such that the process has heard signals $1, 1, 0$ in rounds $t - 2, t - 1, t$, respectively. It then transmits in round $t + 1$ and starts the selection subroutine in round $t + 2$. ∎

By combining Lemma 4.1 with the results from Section 3 and with the existing no-deadlock deterministic algorithms of [8], we obtain the following two conclusions.

Corollary 4.2. *There exists a randomized algorithm with expected makespan* $O(\log n + \log(1/\varepsilon))$ *solving the* ε*-mutual-exclusion problem in the model in which collision detection is available, and a randomized algorithm with makespan* $O(\log n \cdot \log(1/\varepsilon))$ *in the KN model.*

Corollary 4.3. *There exists a deterministic algorithm with makespan $O(\log n)$ solving the mutual exclusion problem in the model in which collision detection is available and a deterministic algorithm with makespan $O(n \log^2 n)$ in the KN model.*

References

[1] H. Attiya, J. Welch, *"Distributed Computing,"*, 2004, John Wiley and Sons, Inc.

[2] R. Bar-Yehuda, O. Goldreich, A. Itai, On the time complexity of broadcast in radio networks: an exponential gap between determinism and randomization, *Journal of Computer and System Sciences*, **45** (1992) 104–126.

[3] M.A. Bender, M. Farach-Colton, S. He, B.C. Kuszmaul, C.E. Leiserson, Adversarial contention resolution for simple channels, in *Proceedings, 17th Annual ACM Symposium on Parallel Algorithms (SPAA)*, 2005, pp. 325–332.

[4] J. Capetanakis, Tree algorithms for packet broadcast channels. *IEEE Transactions on Information Theory* **25** (1979) 505–515.

[5] B.S. Chlebus, L. Gasieniec, D.R. Kowalski, T. Radzik, On the wake-up problem in radio networks, in *Proceedings, 32nd International Colloquium on Automata, Languages and Programming (ICALP)*, 2005, pp. 347–359.

[6] B.S. Chlebus, D.R. Kowalski, M.A. Rokicki, Adversarial queuing on the multiple-access channel, in *Proceedings, 25th ACM Symposium on Principles of Distributed Computing (PODC)*, 2006, pp. 92–101.

[7] A.E.F. Clementi, A. Monti, R. Silvestri, Selective families, superimposed codes, and broadcasting on unknown radio networks, in *Proceedings, 12th Annual ACM-SIAM Symp. on Discrete Algorithms (SODA)*, 2001, pp. 709—718.

[8] J. Czyzowicz, L. Gasieniec, D.R. Kowalski, A. Pelc, Consensus and mutual exclusion in a multiple access channel, in *Proceedings, 23rd International Symposium on Distributed Computing (DISC)*, 2009, pp. 512–526.

[9] L. Gasieniec, A. Pelc, D. Peleg, The wakeup problem in synchronous broadcast systems, *SIAM Journal on Discrete Mathematics*, **14** (2001) 207–222.

[10] L.A. Goldberg, M. Jerrum, S. Kannan, M. Paterson, A bound on the capacity of backoff and acknowledgment-based protocols, *SIAM Journal on Computing* **33** (2004) 313–331.

[11] A.G. Greenberg, S. Winograd, A lower bound on the time needed in the worst case to resolve conflicts deterministically in multiple access channels. *J. ACM* **32** (1985) 589–596.

[12] T. Jurdzinski, M. Kutylowski, J. Zatopianski, Efficient algorithms for leader election in radio networks, in *Proceedings, 21st ACM Symposium on Principles of Distributed Computing (PODC)*, 2002, pp. 51–57.

[13] T. Jurdziński, G. Stachowiak, Probabilistic algorithms for the wakeup problem in single-hop radio networks, in *Proceedings, 13th International Symposium on Algorithms and Computation (ISAAC),* 2002, LNCS 2518, pp. 535–549.

[14] D.R. Kowalski, On selection problem in radio networks, in *Proceedings, 24th ACM Symposium on Principles of Distributed Computing (PODC)*, 2005, pp. 158–166.

[15] Y. Kushilevitz, Y. Mansour, An $\Omega(D \log(N/D))$ lower bound for broadcast in radio networks, *SIAM Journal on Computing* **27** (1998) 702–712.

[16] N.A. Lynch, Distributed Algorithms, Morgan Kaufmann Publ., Inc., 1996.

[17] K. Nakano, S. Olariu, Uniform leader election protocols for radio networks, *IEEE Transactions on Parallel Distributed Systems* **13** (2002) 516–526.

[18] B.S. Tsybakov, V.A. Mikhailov, Free synchronous packet access in a broadcast channel with feedback, *Prob. Inf. Transmission* **14** (1978) 259–280. (Translated from Russian original in *Prob. Peredach. Inf.*, 1977.)

[19] D.E. Willard, Log-logarithmic selection resolution protocols in a multiple access channel, *SIAM Journal on Computing* **15** (1986) 468–477.